НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»
на тему: «Автоматизована система управління постановкою та контролем виконання задач для будівельно-інженерних компаній»

Виконала:
студентка IV курсу, групи КП-51
Голяченко Анастасія Миколаївна _____

Керівник:
Ст. викладач кафедри ПЗКС, к.т.н.,
Люшенко Л.А. _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н.,
Онай М.В. _____

Рецензент:
Доцент кафедри ММСА ІПСА, к.ф.-м.н., доц.
Шубенкова І.А. _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2019 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2018 р.

ЗАВДАННЯ
на дипломний проект студенту
Голяченко Анастасії Миколаївні

1. Тема проекту «Автоматизована система управління постановкою та контролем виконання задач для будівельно-інженерних компаній», керівник проекту Люшенко Леся Анатоліївна, к.т.н., старший викладач, затверджені наказом по університету від «22» травня 2019 р. №1331-С

2. Термін подання студентом проекту «19» червня 2019 р.

3. Вихідні дані до проекту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - аналіз існуючих рішень поставленої задачі;
   - обґрунтування вибору засобів реалізації;
   - розроблення WEB-застосунку "DIP";
   - аналіз розробленого WEB-застосунку "DIP".

5. Перелік обов'язкового графічного матеріалу:
   - діаграма використання (креслення);
   - схема бази даних (креслення);
   - інтерфейс користувача (плакат);

– архітектура системи (плакат).

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	15.09.2018	
2.	Розроблення та узгодження технічного завдання	30.10.2018	
3.	Аналіз аналогів системи та розробка вимог	15.12.2018	
4.	Проектування архітектури системи	01.01.2019	
6.	Аналіз та вибір засобів реалізації.	15.01.2019	
8.	Створення структури бази даних.	15.02.2019	
9.	Розробка інтерфейсу користувача	01.03.2019	
10.	Розробка системи	01.04.2019	
11.	Тестування системи, виправлення недоліків	20.04.2019	
12.	Підготовка матеріалів першого розділу дипломного проекту	01.05.2019	
13.	Підготовка матеріалів другого розділу дипломного проекту	10.05.2019	
14.	Підготовка матеріалів третього розділу дипломного проекту	15.05.2019	
15.	Підготовка матеріалів четвертого розділу дипломного проекту	21.05.2019	
17.	Оформлення документації дипломного проекту	24.05.2019	

Студент А.М. Голяченко

Керівник проекту Л.А. Люшенко

АНОТАЦІЯ

Даний дипломний проект присвячений створенню автоматизованої системи управління постановкою та контролем виконання задач для будівельно-інженерних компаній.

Система являє собою WEB-застосунок із мобільною WEB-версією, призначений для систематизації усіх процесів всередині будівельно-інженерної компанії. Він складається із відповідних модулів: робота із завданнями, редагування профіля користувача, розсилка повідомлень про оновлення (через пошту та Telegram), панель адміністратора (управління проектами та користувачами, побудова індивідуальної ієрахічної структури компанії, журнал змін у системі, генерація звітів), журнал пропозицій та доступ до Державних Будівельних Норм. Система передбачає чотири типи користувачів із різними правами доступу. Лише авторизований користувач може користуватися системою та має доступ до різних функцій залежно від його типу.

У даному дипломному проекті було розроблено: архітектуру WEB-застосунку та бази даних, алгоритм створення ієрархічної структури, процедуру здійснення генерації та перевірки завдань відповідно до рівнів, модуль розсилки оновлень через декілька каналів комунікацій, панель адміністратора, функціонування журналів та інтерфейс користувача.

ABSTRACT

This diploma project deals with the development of the automated system of administration and control of the carrying outcomes for construction engineering companies.

The system is a WEB-application with a mobile WEB-version, designed to systematize all processes within the construction engineering company. It consists of the following modules: work with tasks, editing a user's profile, sending update messages (via e-mail and Telegram), admin panel (project and user management, building an individual company's heritage structure, logging changes in the system, generating reports), a log of proposals and access to State Building Standards. The system provides four types of users with different access rights. Only an authorized user can use the system and has access to various functions depending on its type.

In this project were developed: WEB application architecture and databases, the algorithm for creating a hierarchical structure, the procedure for generating and verifying tasks in accordance with the levels, the module for posting updates through several communication channels, the admin panel, the operation of the logs and the user interface.

ДП.045440-01-90 Автоматизована система управління постановкою та контролем виконання задач для будівельно-інженерних компаній. Відомість проекту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проекту		
ДП.045440-02-91	Автоматизована система	5	
	управління постановкою		
	та контролем виконання		
	задач для будівельно-		
	інженерних компаній.		
	Технічне завдання		
ДП.045440-03-81	Автоматизована система	61	
	управління постановкою		
	та контролем виконання		
	задач для будівельно-		
	інженерних компаній.		
	Пояснювальна записка		
ДП.045440-04-51	Автоматизована система	4	
	управління постановкою		
	та контролем виконання		
	задач для будівельно-		
	інженерних компаній.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Автоматизована система	21	
	управління постановкою		
	та контролем виконання		
	задач для будівельно-		
	інженерних компаній.		
	Керівництво користувача		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-06-99	Автоматизована система	1	
	управління постановкою		
	та контролем виконання		
	задач для будівельно-		
	інженерних компаній.		
	Діяльність користувача у		
	системі. Діаграма		
	використання		
ДП.045440-07-99	Автоматизована система	1	
	управління постановкою		
	та контролем виконання		
	задач для будівельно-		
	інженерних компаній.		
	Архітектура системи.		
	Схема бази даних		
ДП.045440-08-98	Автоматизована система	1	
	управління постановкою		
	та контролем виконання		
	задач для будівельно-		
	інженерних компаній.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

АВТОМАТИЗОВАНА СИСТЕМА УПРАВЛІННЯ ПОСТАНОВКОЮ ТА КОНТРОЛЕМ ВИКОНАННЯ ЗАДАЧ ДЛЯ БУДІВЕЛЬНО-ІНЖЕНЕРНИХ КОМПАНІЙ

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проекту:

_____ Л.А. Люшенко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ А.М. Голяченко

2018

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Автоматизована система управління постановкою та контролем виконання задач для будівельно-інженерних компаній.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розроблюване програмне забезпечення у вигляді WEB-застосунку із мобільною WEB-версією призначене для систематизації усіх процесів у компаніях та наданні зручного інструменту для роботи працівникам будівельно-інженерних компаній.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Система має відповідати наступним вимогам:

- забезпечення функціонування ієрархічної системи всередині кожної окремої компанії, що використовує даний застосунок, а саме:
  - можливість керування інформацією про працівників;
  - надання вказівок конкретним працівникам та контроль їх виконання;
  - зворотній зв'язок із керівничим складом компанії (журнал пропозицій);

- ведення відповідних журналів: із зворотнім зв’язком та проблемами, що виникають (журнал пропозицій) та про будь-які зміни у системі (журнал змін);
- надання доступу до ДБН;
- підтримання відповідного рівня безпеки системи та її швидкодії;
- зручний інтерфейс.

5. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

- пояснювальна записка;
- програма та методика тестування;
- керівництво користувача;
- креслення:
    - «Автоматизована система управління постановкою та контролем виконання задач для будівельно-інженерних компаній. Діаграма використання»;
    - «Автоматизована система управління постановкою та контролем виконання задач для будівельно-інженерних компаній. Схема бази даних».

6. ЕТАПИ ПРОЕКТУВАННЯ

Вивчення літератури за тематикою проект.....15.09.2018

Розроблення та узгодження технічного завдання.....30.10.2018

Аналіз аналогів системи та розробка вимог.....15.12.2019

Проектування архітектури системи.....01.01.2019

Аналіз та вибір засобів реалізації.....15.01.2019

Створення структури бази даних.....15.02.2019

Розробка інтерфейсу користувача 01.03.2019

Розробка системи.....01.04.2019

Тестування системи, виправлення недоліків.....20.04.2019

Підготовка матеріалів першого розділу.....01.05.2019

Підготовка матеріалів другого розділу.....10.05.2019

Підготовка матеріалів третього розділу.....15.05.2019

Підготовка матеріалів четвертого розділу.....21.05.2019

Оформлення документації дипломного проекту.....24.05.2019

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

АВТОМАТИЗОВАНА СИСТЕМА УПРАВЛІННЯ ПОСТАНОВКОЮ ТА КОНТРОЛЕМ ВИКОНАННЯ ЗАДАЧ ДЛЯ БУДІВЕЛЬНО-ІНЖЕНЕРНИХ КОМПАНІЙ

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проекту:

_____ Л.А. Люшенко

Виконавець:

Нормоконтроль:

_____ М.В. Онай

_____ А.М. Голяченко

2019

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ДБН – *державні будівельні норми* – нормативно-правові акти, затверджені центральним органом виконавчої влади з питань будівництва та архітектури. Державні будівельні норми охоплюють усі галузі народного господарства України.

НДІ – *нормативно-довідкова інформація* – компонент корпоративної інформації, що є основою для уніфікації і нормалізації даних, які супроводжують бізнес-процеси, що протікають, а також регламентацію діяльності організації.

Інтерфейс користувача (англ. user interface, UI, *дружній інтерфейс*) – засіб зручної взаємодії користувача з інформаційною системою.

ЕВМ – *електронна вичислювальна машина* – комп'ютер.

БД (англ. Database, *база даних*) – сукупність даних, систематизована та представлена у певному вигляді, щоб ці матеріали могли бути знайдені та опрацьовані за допомогою ЕВМ.

СКБД (англ. Database Management System, скор. DBMS) – *система керування базами дани* – сукупність програмних та лінгвістичних засобів загального або спеціального призначення, що забезпечують управління створенням або використанням БД. Система управління базами даних представляє собою комплекс програм, що дають можливість створювати та видаляти БД, маніпулювати даними у БД, вносити зміни у БД, тощо.

ПЗ (англ. software) – *програмне забезпечення* – программа або множина програм, що використовуються для управління комп'ютером.

Архітектура ПЗ (англ. software architecture, *архітектура ПЗ*) – сукупність рішень, що організовують систему програми ізсередини.

Фреймворк (англ. framework) – *каркас, структура* – заготовки, шаблони для програмної платформи, що визначають структуру системи програми. Фреймворк – це ПЗ, що облегшує та прискорює розробку та об'єднання різних модулей програмного проекту.

Django – фреймворк, за допомогою якого створюються WEB-застосунки мовою програмування Python, використовує шаблон проектування MVC.

MVC (англ. Model-View-Controller) – *модель-представлення-контролер* – схема, що розділяє дані застосунку, інтерфейс користувача та управляючу логіку на три компоненти – модель, представлення та контролер відповідно. Таким чином модифікація кожного з компонентів може відбуватися незалежно один від одного.

Ajax (англ. Asynchronous Javascript and XML) – асинхронний Javascript та XML – підхід до побудови інтерактивних користувацьких інтерфейсів WEB-застосунків, що представляє собою "фоновий" обмін даних між браузером та WEB-сервером, дані на WEB-сторінці оновлюються автоматично без перезавантаження сторінки.

Javascript – мультипарадигмальна мова програмування.

SQL (англ. Structured Query Language) – *мова структурованих запитів* – декларативна мова програмування, яка застосовується для створення, модифікації та управління даними в реляційній базі даних, керованій відповідною системою керування базами даних.

C – компільована статично типізована мова програмування загального призначення, розроблена в 1969-1973 роках співробітником Bell Labs Денніс Рітчі як розвиток мови Бі.

libpq – це бібліотека, інтерфейс PostgreSQL для програмування додатків на мові C. Бібліотека libpq містить набір функцій, використовуючи які клієнтські програми можуть передавати запити серверу PostgreSQL і приймати результати цих запитів.

XML (англ. e**x**tensible **m**arkup **l**anguage) – *розширювана мова розмітки* – специфікація XML, що описує XML-документи та поведінку XML-процесорів.

API (англ. Application Programming Interface) – *програмний інтерфейс застосунку* - опис способів (набір класів, процедур, функцій, структур або констант), якими одна комп'ютерна програма може взаємодіяти з іншою програмою.

ORM (англ. Object-Relational Mapping) – *об'єктно-реляційне відображення, або перетворення* – технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи «віртуальну об'єктну базу даних». Існують як пропрієтарні, так і вільні реалізації цієї технології.

Bootstrap – вільний набір інструментів для створення сайтів і веб-додатків. Включає в себе HTML- і CSS-шаблони оформлення для типографіки, веб-форм, кнопок, міток, блоків навігації та інших компонентів веб-інтерфейсу, включаючи JavaScript-розширення.

HTML (англ. HyperText Markup Language) – *мова гіпертекстової розмітки* – стандартизована мова розмітки документів у Всесвітній павутині. Більшість веб-сторінок містять опис розмітки на мові HTML (або XHTML). Мова HTML інтерпретується браузерами, отриманий в результаті інтерпретації форматований текст відображається на екрані монітора комп'ютера або мобільного пристрою.

Psycopg2 – сумісний із Python драйвер PostgreSQL, який активно розвивається. Він призначений для багатопоточних застосунків і управляє власним пулом підключень. Можливості даного адаптера полягають у тому, що при використанні типу даних PostgreSQL, Psycopg2 автоматично перетворить результат та переформатує його у відповідний тип даних у Python.

GNU GPL (англ. GNU General Public License) – *універсальна громадська ліцензія GNU, або відкрита ліцензійна угода GNU* – ліцензія на вільне програмне забезпечення, створена в рамках проекту GNU у 1988 р., за якою автор передає програмне забезпечення до суспільної власності.

Telegram – багатоплатформний месенджер, що дозволяє обмінюватися повідомленнями і медіафайлами багатьох форматів. Використовується пропріетарна серверна частина із закритим кодом, що працює на потужностях декількох компаній США і Німеччини та кілька клієнтів з відкритим вихідним кодом, в тому числі під ліцензією GNU GPL.

ВСТУП

У наші дні розвиток інформаційних технологій (ІТ) досяг такого рівня, що вже повсюди супроводжує життя сучасної людини. Кожен користується такими популярними гаджетами як телефон, ПК чи планшет. Вони допомагають зі швидким пошуком необхідної інформації, прискорюють виконання процесів та зберігають усі потрібні дані у структурованому та зручному форматі.

Однак, ще не у всіх галузях життя ІТ розглядаються у форматі обов'язкового інструменту для роботи. Велика частина будівельно-інженерного ринку України не користується спеціально розробленим ПЗ, де б була розміщена уся проектно-кошторисна документація, був би вільний оф-лайн доступ до ДБН та водночас із цим була можливість управління постановкою та контролем виконання задач.

Даний дипломний проект присвячено розробленню програмного забезпечення для будівельно-інженерних компаній під назвою "DIP". Воно буде виконано у вигляді WEB-застосунку та матиме мобільну WEB-версію для більш зручної роботи.

Дане програмне забезпечення спрямоване на збільшення ефективності кожного співробітника будівельно-інженерної компанії, у якій воно застосовується.

Метою даного проекту є зменшення часових затримок, які утворюються під час з'ясування та вирішення дрібних питань, оптимізація паралельних непотрібних завдань та процесів, зниження кількості осіб керуючого складу компанії, усунення помилок, що відбуваються під час роботи на будівництві чи взаємодії із керівництвом, або іншими службами при отриманні відповідних дозвільних документів, тощо.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ПОСТАВЛЕНОЇ ЗАДАЧІ

1.1. Загальні положення та аналіз предметної області

Розвиток будівельно-інженерної галузі на території України зараз досягає все більших показників. За останні три роки (2016 – 2018 рр.) ринок значно розширився, з'явилося багато нових компаній, що підтверджує зростання даної галузі у країні. Відповідно відбувається зростання конкуренції та темпу виконання роботи, що викликає нові вимоги до будівельних компаній у галузі.

У рамках даного дипломного проекту було проведено опитування серед топ-менеджерів будівельно-інженерних компаній і всі вони відповіли, що витрачають неефективно свій робочий час. Серед них близько 60% зазначили, що витрачають більше 3 годин на тиждень, та 40% – від однієї до трьох годин на тиждень на з'ясування та вирішення дрібних питань, що створює значні часові затримки у роботі. Також серед найважливіших проблем, що виникають, вони зазначили такі:

- відсутність чіткого плану дій та розподіл їх на відповідальних осіб;
- велика кількість осіб керуючого складу та наявність паралельних непотрібних завдань;
- помилки, що допускаються через незнання будівельних норм;
- велика кількість паперів завжди із собою;
- неструктурованість документації та важкий пошук по ній.

Проаналізувавши дані опитування, запропоновано заходи, які дозволять зменшити витрати часу, а саме:

- впровадження системи, яка контролює надання та виконання задач у відповідний термін;
- наявність структурованої проектно-кошторисної документації у мобільному пристрпої;
- можливість офлайн доступу до ДБН;

- впровадження журналів контролю змін у документації та проблем, що виникали під час роботи;
- автоматичне розподілення завдань відповідно до посад на етапі оформлення дозвільної документації перед початком будівництва.

Розроблюване програмне забезпечення у вигляді WEB-застосунку із мобільною WEB-версією спрямоване на вирішення проблем за допомогою впровадження перелічених вище заходів.

1.2. Аналіз аналогічних застосунків

Було проаналізовано ринок аналогічних застосунків, що використовуються будівельно-інженерними компаніями. Серед найпопулярніших було визначено “АВК-5” та “Гранд-смета”.

“АВК-5”

Програмний комплекс «Автоматизований випуск кошторисів – 5», починаючи з версії 2.8.0, є подальшим розвитком програмних комплексів АВК (АВК-1, АВК-2, АВК-3), створених фірмою-розробником "НПФ АВК" у 1994 р.

Програмний комплекс «АВК-5» призначений для автоматизації визначення вартості будівництва – нового, реконструкції, капітального ремонту чи технічного переоснащення. На сьогоднійшній день він орієнтований на вимоги національного стандарту ДСТУ Б Д.1.1–1:2013 «Правила визначення вартості будівництва» і призначений для автоматизованого випуску ресурсної вартісної документації, що викорситовується у будівництві:

- у інвесторських кошторисах (підсистема «Кошторисні документи»);
- у договірних цінах контрактів (підсистема «Договірна ціна»);
- у виконанні будівельних робіт (підсистема «Підрядник»).

Крім того у ньому реалізовані алгоритми, що дозволяють визначати вартість будівельних робіт згідно відомчих нормативних документів:

- Міністерства промислової політики України;

- Міністерства транспорту та зв'язку України;
- Державного агентства автомобільних доріг України;
- Житлово-комунального господарства України.

Програмне забезпечення надає можливість користувачеві створювати власні регіональні та будівельні норм, заносячи їх до НДІ. Наявність розвиненої пошукової системи спрощує пошук необхідних записів НДІ і скорочує час розробки кошторисної документації [1].

Система “АВК-5” є додатком для комп’ютера, який потрібно завантажувати та встановлювати, тому має чіткі системні вимоги:

- операційна система – Windows XP і вище;
- оперативна пам’ять – не менше 1 Гб;
- тактова частота процесору – 1 ГГц і вище;
- об’єм вільного дискового простору – не менше 2 Гб;
- застосунок не працює на віртуальних машинах та у термінальних режимах.

“АВК-5” має достатньо простий, зрозумілий інтерфейс для роботи, проте дизайн значно застарів (рис. 1).

“Гранд-смета”

“Гранд-смета” – програмний комплекс для складання і перевірки кошторисних розрахунків, а також створення актів виконаних робіт по різним формам довідок, складання всіх видів кошторисної документації для визначення вартості будівництва. Розробник – група компаній «ГРАНД».

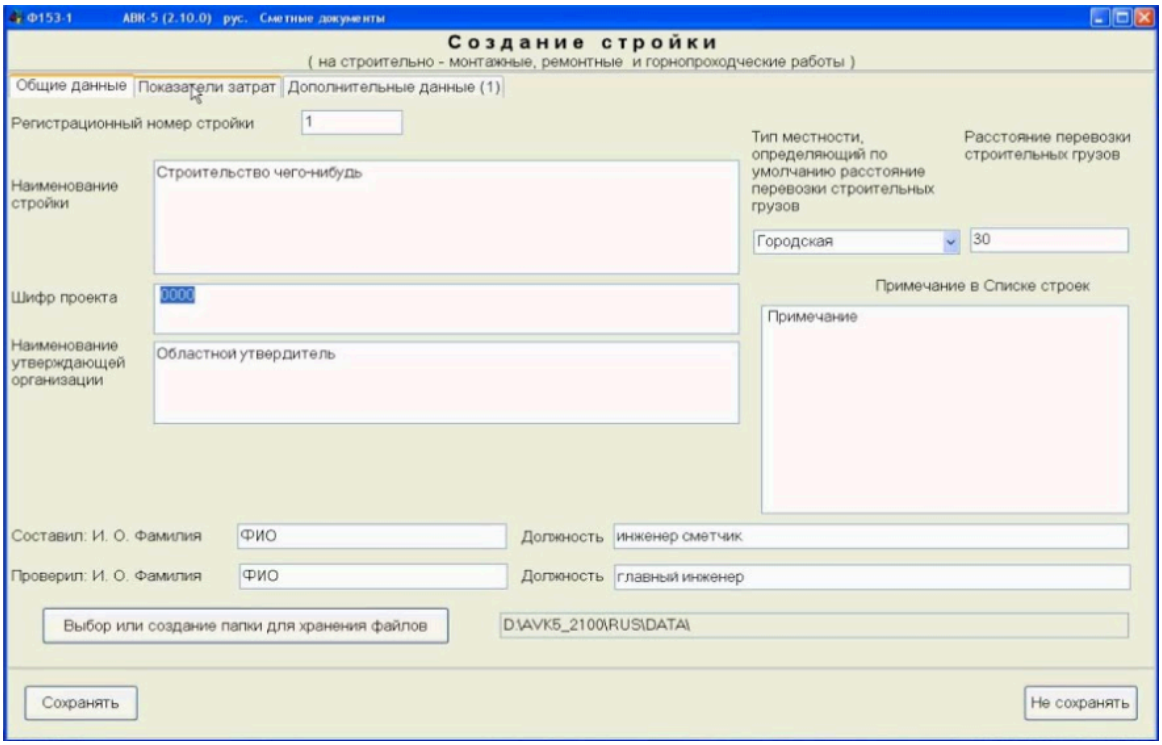


Рис. 1. Сторінка «Створення будівництва» у програмі «АВК-5»

Основними функціями даної системи є:

- зведення кошторисного розрахунку вартості будівництва, з можливістю автоматичного створення на основі локальних, об'єктних розрахунків, складених у програмі;
- розрахунок показника одиничної вартості;
- автоматичне формування відомості обсягів робіт за кошторисом на основі даних локального кошторису;
- автоматичне створення ресурсної відомості на основі даних локального кошторису;
- генерація звіту про витрату матеріалів;
- автоматичний розрахунок підсумкової вартості по розділах і за кошторисом [2].

“Гранд-смета” є програмним забезпеченням, що потрібно завантажувати та установлювати на комп’ютер. Системні вимоги:

- операційна система – Windows 7 та вище;
- оперативна пам'ять – 2 Гб і більше (рекомендується 4 Гб);

- тактова частота процесору – не нижче 2ГГц;
- об’єм вільного дискового простору – не менше 300 Мбайт + 200 Мб для кожної нормативної бази;
- MS Office 2010 та вище;
- OpenOffice 4.0 та вище.

«Гранд-смета» має простий інтерфейс за структурою нагадуючий Microsoft Excel (рис. 2).

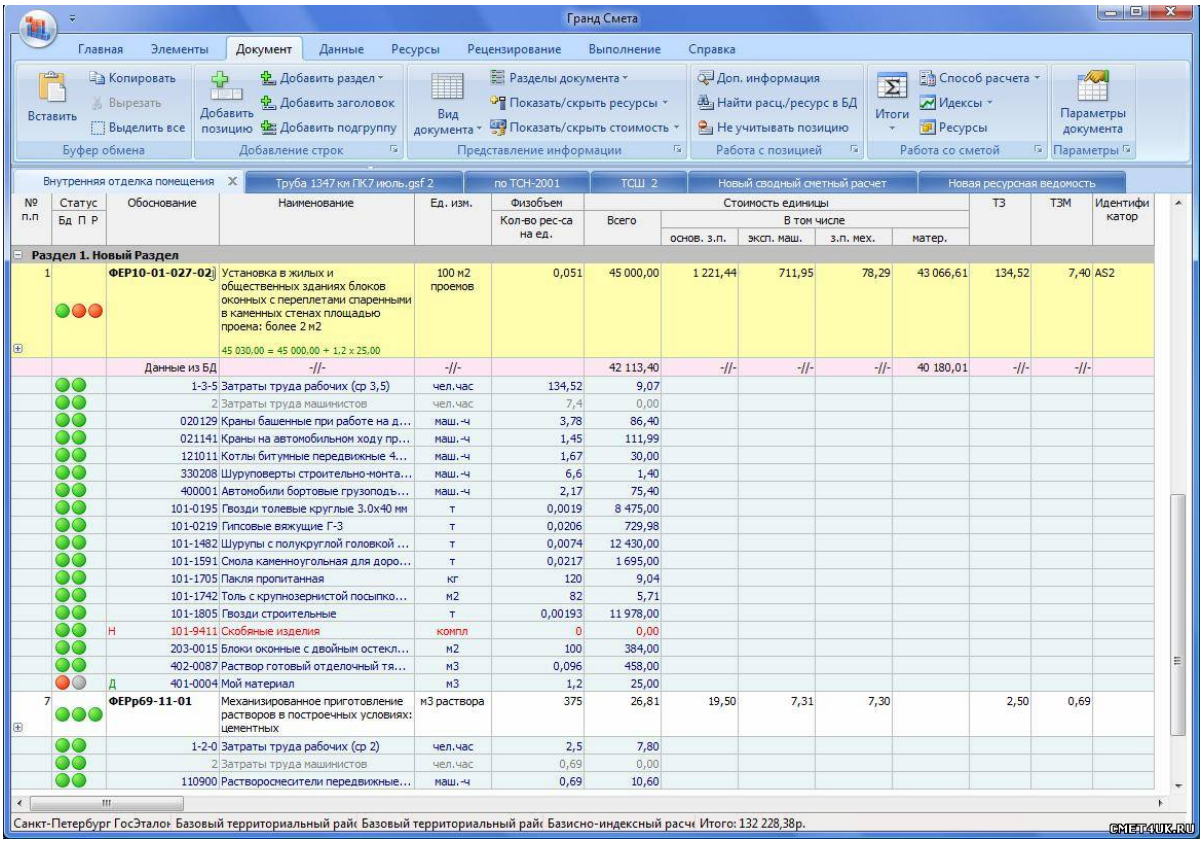


Рис. 2. Розділ «Документ» у програмі «Гранд-смета»

1.3. Актуальність розробки WEB-застосунку “DIP”

У розділі із аналізом аналогічних застосунків було наведено приклади систем, що автоматизують процес створення документації, зокрема кошторисних розрахунків вартості будівництва. У даному дипломному проекті розроблюване програмне забезпечення також призначене для автоматизації процесу ведення проектно-кошторисної документації, а також має частину функцій, що призначені для управління постановкою та

контролем задач, ведення журналів змін у документації та проблем, що виникають під час роботи, надання офлайн доступу до ДБН.

Розроблюване програмне забезпечення охоплює більшу кількість напрямків роботи будівельно-інженерної компанії аніж наведені приклади. Окрім вище згаданого, воно створюється у форматі WEB-застосунку із мобільною WEB-версією. Це робить можливим використання застосунку на мобільних пристроях. Аналогічні комплексні програми зараз застосовуються лише у небагатьох великих компаніях та були розроблені індивідуально для них. Відповідно їх вартість дуже значна і лише найбільші компанії можуть їх собі дозволити.

Метою даного програмного забезпечення “DIP” є розроблення універсальної та доступної програми для будівельно-інженерних компаній із різними структурами посад та напрямами проектів. Розроблене програмне забезпечення має:

- забезпечувати функціонування ієрархічної системи всередині кожної окремої компанії, а саме: можливість керування інформацією про працівників, надання вказівок конкретним працівникам або їх групам та контроль їх виконання, зворотній зв’язок із керівничим складом компанії;
- вести відповідні журнали: із записами про проблеми, що виникають під час роботи, та будь-які зміни у системі;
- надавати доступ до проектної документації та ДБН;
- підтримувати відповідне шифрування даних задля підтримання інформаційної безпеки.

1.4. Загальні вимоги до системи

На основі аналізу розглянутих існуючих аналогів та попереднього опитування цільової аудиторії було розроблено відповідні загальні вимоги до системи: функціональні та нефункціональні.

Розроблене програмне забезпечення у вигляді WEB-застосунку із мобільною WEB-версією для інженерно-будівельних компаній має відповідати таким функціональним вимогам:

- забезпечення різних рівнів доступу та прав для користувачів – адміністратор, топ-менеджер, менеджер, звичайний користувач (див. підрозділ 3.1. “Загальний опис системи”);
- система передачі та контролю виконання завдань:
  - розподілення автоматизованих завдань і створення завдань у ручному режимі;
  - функціонування коментарів до кожного окремого завдання із можливістю згадування користувача у форматі “@ім’я_користувача” для забезпечення комунікативного каналу між працівниками;
  - можливість редагування усіх елементів завдання користувачами типу “Менеджер”, “Топ-менеджер” та “Адміністратор”;
- ведення журналу пропозицій:
  - можливість додавання пропозицій користувачами;
  - автоматична генерація звітів щодо пропозицій за певний період часу;
- ведення журналу змін:
  - автоматизоване заповнення журналу змін при будь-яких діях користувачів, що стосуються завдань чи особистої інформації, оновлення ДБН;
  - можливість генерації звітів щодо змін внесених у завдання проекту чи особисту інформацію акаунту користувачів, оновлення у ДБН за певний період часу;
- робота із власною сторінкою користувача:
  - зміна фото, ім’я користувача у форматі “@ім’я_користувача”, паролю, електронної пошти;

- додавання інформації про своє ім’я у месенджері Telegram для активізації розсилки повідомлень про оновлення.

Розроблене ПЗ має відповідати таким нефункціональним вимогам:

- WEB-застосунок із мобільною WEB-версією;
- зручний для користувача інтерфейс.

Вимоги до якості розроблюваного ПЗ:

- відповідне шифрування даних для забезпеченння конфіденційності даних проекту;
- швидкісні характеристики;
- безперервний доступ для користувачів.

1.5. Вимоги до безпеки системи

Оцінка ризиків програмного забезпечення

Оцінка ризиків є важливим етапом у розробленні програмного застосунку, оскільки він допомагає зробити певні висновки щодо безпеки продукту та створити необхідні заходи для зменшення виявлених ризиків.

На основі розглянутих ризиків можна сформувати відповідні наслідки, до яких вони можуть призвести. У разі недопустимості наслідків створюються вимоги до безпеки системи, які мають бути виконані для стабільної та безпечної роботи системи.

Під час роботи були розглянуті можливі уразливості системи, сформульовані можливі загрози, що виникають, і відповідні наслідки. Серед найімовірніших ризиків, що можуть виникнути під час роботи системи, можна назвати:

- відсутність захисту від SQL-ін’єкцій;
- відсутність перевірки введених даних;
- відсутність шифрування даних, що зберігаються.

У табл. 1 наведені ризики розроблюваного програмного забезпечення.

Таблиця 1

Результати ідентифікування ризику безпеки програмного забезпечення

№ з/с	Програмне забезпечення	Уразливість	Загроза	Наслідки
1.1	Програмне забезпечення “DIP”	Відсутність захисту від SQL-ін’єкцій	Зміна чи видалення даних у базі даних	Порушення цілісності даних, що зберігаються у системі
1.2			Викрадення даних	Втрата даних
2.1		Відсутність перевірки введених даних	Доступ до конфіденційних даних	Порушення конфіденційності даних
2.2			Проблеми у роботі застосунку	Нестабільна робота застосунку
3.1		Відсутність шифрування даних, що зберігаються	Доступ до конфіденційних даних	Порушення конфіденційності даних
3.2			Викрадення конфіденційних даних	Втрата конфіденційних даних

Визначення вимог до безпеки системи

Наведені ризики та їх наслідки є недопустимими у розроблюваній системі, тому було прийнято рішення створення вимог до безпеки.

За умови виконання розроблених вимог до безпеки вищезгадані ризики будуть усунені.

У табл. 2 наведені відповідні вимоги до безпеки програмного забезпечення, що були прийняті після аналізу.

Таблиця 2

Оброблені ризики безпеки програмного забезпечення

№ з/с	Програмне забезпечення	Уразливість	Загроза	Вимоги до безпеки
1.1	Програмне забезпечення “DIP”	Відсутність захисту від SQL-ін’єкцій	Зміна чи видалення даних у базі даних	Забезпечення захисту системи від SQL-ін’єкцій при вводі даних
1.2			Викрадення даних	
2.1		Відсутність перевірки введених даних	Доступ до конфіденційних даних	Забезпечення перевірки введених даних перед подальшим їх збереженням та використанням
2.2			Проблеми у роботі застосунку	
3.1		Відсутність шифрування даних, що зберігаються	Доступ до конфіденційних даних	Шифрування даних, що зберігаються у системі
3.2			Викрадення конфіденційних даних	

1.6. Висновки до розділу

На сьогоднішній день у більшості компаній на цьому ринку відсутнє програмне забезпечення, яке дозволяє ефективно управляти роботою проектно-будівною компанією. Тому виникає значне навантаження на співробітників компаній, губляться документи, випадкові помилки, тощо. Через це керівництво компанії збільшує кількість осіб керуючого складу, але у результаті створюються лише паралельні непотрібні процеси і це не вирішує проблему ефективного управління компанією.

Розроблюване програмне забезпечення у вигляді WEB-застосунку із мобільною WEB версією призначене для систематизації усіх процесів у компанії та надання зручного інструменту для роботи співпрацівникам будівельно-інженерних компаній.

Існуючі комплексні аналоги розробляються виключно у індивідуальному порядку для великих будівельно-інженерних компаній і коштують дуже дорого.

Відповідно до вищерозглянутих проблем, найпоширеніших у сучасній будівельно-інженерній галузі, та способів їх вирішення, розроблюване програмне забезпечення має бути представлене у вигляді WEB-застосунку із мобільною WEB версією, забезпечувати функціонування ієрархічної системи всередині компанії, включаючи надання та контроль виконання завдань, доступ до проектно-кошторисної документації та ДБН та ведення відповідних журналів.

Програмне забезпечення “DIP” має бути універсальним та доступним для більшості компаній, що є головною його перевагою.

Також було визначено функціональні і нефункціональні вимоги до розроблюваного програмного забезпечення і зазначені вимоги до якості системи.

Окрім цього було проведено аналіз можливих ризиків відносно безпеки даної системи, проведено оцінку ризиків та визначено відповідні наслідки за умови їх настання. Оскільки усі наведені ризики та їх наслідки є недопустимими у розроблюваному програмному забезпеченні, було прийнято рішення створення вимог до безпеки, виконання яких усуває можливість виникнення вищезгаданих ризиків.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Обґрунтування вибору мови програмування

Оскільки розроблюване програмне забезпечення для будівельно-інженерних компаній має бути представлене у вигляді WEB-застосунку вибір мови програмування був заснований на огляді найпопулярніших мов програмування для WEB-застосунків.

Мова програмування має відповідати таким вимогам:

- мати широке застосування, постійно розвиватися та оновлюватися;
- активний ріст кількості відкритих бібліотек для використання;
- наявність зручного кросплатформного MVC.

Огляд мови програмування Python

Python – високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника і читання коду. Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий обсяг корисних функцій [3].

До основних архітектурних рис належать: динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень, високорівневі структури даних. Підтримується розбиття програм на модулі, які, в свою чергу, можуть об'єднуватися в пакети. Python є мультипарадигмальною мовою програмування та підтримує структурне, об'єктно-орієнтоване, функціональне, імперативне і аспектно-орієнтоване програмування [4].

Переваги Python:

- значна швидкість створення програмного забезпечення через простий синтаксис мови Python;
- наявність великої кількості відкритих бібліотек та постійний розвиток цієї мови програмування;
- легкість редагування коду та його зрозумілість;

- наявність MVC – Django та зручність його використання на різних ОС [5].

Головним недоліком Python є низька швидкодія, однак, цей недолік компенсується зменшенням часу розробки програми. У середньому, програма, написана на Python, в 2-4 рази компактніша, ніж її аналог на C# або Java.

Огляд мови програмування C#

C# – об'єктно-орієнтована мова програмування. Вона відноситься до сім'ї мов з C-подібним синтаксисом, синтаксис найбільш близький до C ++ і Java. Дана мова програмування була створена у 2000 році, автор – Андерс Хейлсберг.

Мова C# має: статичну типізацію, підтримує поліморфізм, перевантаження операторів, делегати, атрибути, події, тощо. C# є мультипарадигмальною мовою програмування та підтримує об'єктно-орієнтоване, узагальнене, процедурне, функціональне, подійно-орієнтоване та рефлексивне програмування.

Переваги C#:

- наявність великої кількості бібліотек та шаблонів;
- доступний та зрозумілий для читання код;
- наявність технології ASP.NET, призначеної для WEB-розробки.

Недоліком C# є те, що дана мова програмування протягом довгого періоду часу підтримувалася лише пристроями с ОС Windows. На даний момент корпорація Microsoft вже відкрила вихідний код .NET та зробила його крос-платформним. Але розробники все одно мають отримувати ліцензоване програмне забезпечення для розгортання проектів, яке є платним для операційних систем, відмінних від Windows.

Огляд мови програмування Javascript

Мова програмування Javascript є реалізацією мови ECMAScript. Вона програмування була створена у 1995 році, автор – Брендан Ейх.

Зазвичай JavaScript використовується у форматі вбудованої мови, яка описує сценарії, що додають інтерактивність WEB-сторінкам.

Основні архітектурні риси Javascript: динамічна слабка типізація, автоматичне керування пам'яттю,прототипне програмування, функції як об'єкти першого класу. JavaScript є мультипарадигмальною мовою програмування та підтримує об'єктно-орієнтоване, узагальнене, функціональне, імперативне, подійно-орієнтоване, аспектно-орієнтоване програмування.

Переваги JavaScript:

- мова постійно розвивається;
- пряме підключення скриптів до HTML коду;
- можливість запуску програм в браузері і на сервері;
- широкий вибір корисних функціональних параметрів.

Головним недоліком використання мови програмування JavaScript є доволі незвичайний формат об’єктної моделі, що ускладнює розробку та подальшу підтримку програмного забезпечення при його ускладненні.

Огляд мови програмування Java

Java – сильно типізована об'єктно-орієнтована мова програмування, розроблена компанією Sun Microsystems (потім придбана Oracle). Дата офіційного випуску – 23 травня 1995 року, автор – Джеймс Гослінг.

Програми на Java транслюються в байт-код Java, який виконується віртуальною машиною Java (JVM) – програмою, що обробляє байтовий код і передає інструкції обладнанню в якості інтерпретатора. Перевагою подібного способу виконання програм є повна незалежність байт-коду від операційної системи і устаткування, що дозволяє виконувати Java-застосунки на будь-якому пристрої, для якого існує відповідна віртуальна машина. Java є мультипарадигмальною мовою програмування та підтримує об'єктно-орієнтоване, структурне, функціональне, імперативне, узагальнене, компонетно-орієнтоване та рефлективне програмування.

Переваги Java:

- стабільність та наявність певної спільноти через довгий період застосування;
- є стандартом для корпоративних обчислювальних систем.

Серед недоліків використання мови програмування Java найчастіше згадується зниження продуктивності через використання концепції віртуальної машини.

Обрана мова програмування

Серед вищеперелічених мов програмування були виявлені такі суттєві недоліки саме для розроблюваного програмного забеспечення:

- для використання мови програмування C# розробники мають отримувати ліцензоване програмне забезпечення для розгортання проектів, яке є платним для операційних систем, відмінних від Windows. Оскільки запланована розробка ведеться на Mac OS цей недолік є критичним.
- Головним недоліком використання мови програмування JavaScript є незвичайний формат об’єктної моделі, що ускладнює розробку та подальшу підтримку програмного забезпечення при його ускладненні. Оскільки для розроблюваного програмного забезпечення планується подальший активний розвиток і внесення багатьох нововведень, мова програмування JavaScript не підходить.
- Варіант використання мови програмування Java також був відкинутий через зниження продуктивності роботи із використанням віртуальної машини.

Отже, визначивши необхідні вимоги до мови програмування для розроблюваного програмного забезпечення, зробивши огляд найпопулярніших мов програмування для розробки WEB-застосунків та проаналізувавши їх, було зроблено вибір у сторону мови програмування

Python. Дана мова має простий та зрозумілий синтаксис, є достатньо молодою та активно розвивається, має широкий вибір відкритих для використання бібліотек. Для написання програмного забезпечення використовуватиметься WEB-фреймворк Django, який є кросплатформеним та зручним інструментом для написання WEB-застосунків на Python [6].

2.2. Обґрунтування вибору системи керування базами даних

Розроблюване програмне забезпечення для будівельно-інженерних компаній у вигляді WEB-застосунку має зберігати велику кількість даних, тому використання СКБД є необхідним. Нижче приведено короткий огляд найпопулярніших СКБД використовуваних для WEB-застосунків із перерахуванням їх переваг та недоліків з огляду на необхідну розробку.

Огляд СКБД MySQL

Тип баз даних – реляційна. Розробник – Oracle Corporation. MySQL є однією з найпопулярніших баз даних для WEB-застосунків. Вона представляє собою безкоштовний пакет програм, який постійно оновлюється, розширюються функціональні можливості та покращується рівень безпеки. Також існують спеціальні платні версії, що призначені для комерційного користування [7].

Дана СКБД дозволяє використовувати різні “движки” для системи зберігання, які дозволяють змінювати функції інструменту і виконувати обробку даних, що зберігаються в різних типах таблиць.

Гнучкість MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці, що підтримують повнотекстовий пошук, так і таблиці, що підтримують транзакції на рівні окремих записів. Завдяки відкритій архітектурі і GPL-ліцензуванню, в даній СКБД постійно з'являються нові типи таблиць.

Вона має простий у використанні інтерфейс, і пакетні команди, які дозволяють зручно обробляти великі обсяги даних. Система надійна і не намагається підпорядкувати собі всі доступні апаратні ресурси.

Переваги використання СКБД MySQL:

- є безкоштовна версія, що пропонує багато функцій;
- документація чітка та зрозуміла;
- легка у встановленні;
- підтримує набір інтерфейсів призначених для користувача;
- Може працювати з іншими базами даних, включаючи DB2 і Oracle.

Недоліки використання СКБД MySQL:

- великі часові затрати на виконання нескладних завдань, в той час, коли інші системи роблять це автоматично (наприклад, створення інкрементних резервних копій);
- відсутня вбудована підтримка XML або OLAP;
- для безкоштовної версії доступна тільки платна підтримка.

Огляд СКБД PostgreSQL

Тип баз даних – об'єктно-реляційна. Розробник: PostgreSQL Global Development Group. PostgreSQL є також безкоштовним та популярним варіантом для використання у WEB-застосунках. Ця СКБД була однією із найперше розроблених, тому на даний момент вона добре розвинена та дозволяє користувачам управляти як структурованими так і неструктурованими даними. Движок даної СКБД може бути розміщений в ряді середовищ, в тому числі віртуальних, фізичних і хмарних [8].

Найостанніші версії здійснюють обробку великих обсягів даних із паралельним збільшенням кількості одночасно працюючих користувачів, а також значно покращений рівень безпеки.

Переваги використання СКБД PostgreSQL:

- є масштабованою і здатна обробляти терабайти даних;
- підтримує формат json;

- має дуже багато визначених функцій;
- доступний ряд інтерфейсів для користувачів.

Недоліком використання СКБД PostgreSQL є доволі складне налаштування конфігурації.

Огляд СКБД Microsoft SQL Server

Тип баз даних – реляційна. Розробник – Microsoft

Microsoft SQL Server є ще однією з найпопулярніших СКБД. У Microsoft SQL Server движок працює на хмарних серверах та локальних, що дає змогу комбінувати типи застосовуваних серверів одночасно [9]. Незабаром після випуску Microsoft SQL сервер 2016, Microsoft адаптувала продукт для операційної системи Linux.

Переваги використання СКБД Microsoft SQL Server:

- продукт дуже простий у використанні;
- поточна версія працює швидко і стабільно;
- движок надає можливість регулювати і відслідковувати рівні продуктивності, які допомагають знизити використання ресурсів;
- наявність візуалізації для мобільних пристроїв.

Недоліки використання СКБД Microsoft SQL Server:

- вагома ціна для юридичних осіб;
- намагається зайняти усі доступні ресурси;
- проблеми з використанням служби інтеграції для імпорту файлів.

Огляд СКБД MongoDB

Тип баз даних – документоорієнтована. Розробник – MongoDB Inc.

MongoDB є безкоштовною СКБД, але існує и комерційна версія. Вона застосовується у додактках як із структурованими даними, так із неструктурованими.

Ядро є дуже гнучким і працює при підключенні бази даних до застосунків через драйвери MongoDB. Існує широкий вибір доступних

драйверів, тому легко знайти той, що буде працювати з необхідною мовою програмування. Движок призначений для обробки різних даних, і реляційних і нереляційних, і може добре справлятися там, де інші движки працюють повільно або безсилі. Останні версії MongoDB мають нову систему движків зберігання, що можна підключити. Документи тепер можуть бути перевірені у процесі оновлення або виконання вставок, також були покращені функції текстового пошуку. Нова здатність часткового індексування підвищує продуктивніть, зменшуючи розмір індексів [10].

Переваги використання СКБД MongoDB:

- продуктивність і простота у використанні;
- підтримка json та інших традиційних документів NoSQL;
- дані будь-якої структури можуть бути збережені чи прочитані швидко і легко.

Недоліки використання СКБД MongoDB:

- SQL не використовується в якості мови запитів;
- інструменти для перекладу SQL-запитів в MongoDB доступні, але їх слід розглядати саме як доповнення;
- погана сумісність із Django.

Обрана СКБД

Серед вищеперелічених СКБД було визначено такі критичні недоліки у відношенні до розроблюваного програмного забезпечення:

- СКБД MySQL значно повільніша за інші СКБД у випадку із простими операціями, що мали б виконуватися автоматично. У випадку розроблюваного програмного забезпечення це є важливим недоліком, оскільки часові затримки можуть призвести до певних наслідків під час будівництва та затримати роботу.
- СКБД Microsoft SQL Server має високу ціну для використання юридичними особами. Оскільки розроблюване програмне

забезпечення планується для виходу на ринок – це є вагомим недоліком.

- СКБД MongoDB має погану сумісність із обраним фреймворком для роботи Django, що значно сповільнить швидкодію застосунку та обмежить можливості.

Для розроблюваного програмного забезпечення було обрано СКБД PostgreSQL. Вона є масштабованою і може швидко обробляти великі об’єми даних, має високий рівень безпеки, що є важливим саме для розроблюваного WEB-застосунку. Також вона може управляти різними типами даних, що необхідно для розробки, адже потрібно зберігати дані як структуровані, так і неструктуровані. Також дана СКБД є продуктивною і водночас простою у використанні, має дуже багато визначених функцій, а також доступний ряд інтерфейсів для користувачів. Дана СКБД є сумісною із Django і офіційно підтримується, що забезпечує високий рівень ефективності та зручності роботи із нею.

2.3. Висновки до розділу

У даному розділі було виконано збір вимог до технологій розробки, а саме мови програмування та системи керування базами даних, з огляду на розроблюване програмне забезпечення у вигляді WEB-застосунку та мобільної WEB версії. Також було проведено огляд найпопулярніших технологій розробки – серед мов програмування: Python, JavaScript, Java, C#; серед СКБД: MongoDB, MySQL, PostgreSQL, Microsoft SQL Server. Було перелічено їх переваги та недоліки.

Серед мов програмування було обрано мову Python через швидкість та зручність розробки, простий синтаксис, наявність розвинутого WEB-фреймворку Django, кросплатформність, постійний розвиток та появу відкритих для використання бібліотек.

Серед СКБД було обрано PostgreSQL, адже вона є високопродуктивною, масштабованою та забезпечує високий рівень безпеки. Також вона може

управляти як структурованими так і неструктурованими даними, що є важливим для розроблюваного застосунку. Дана СКБД є сумісною із Django і офіційно підтримується, що забезпечує високий рівень ефективності та зручності роботи із нею.

3. РОЗРОБЛЕННЯ WEB-ЗАСТОСУНКУ “DIP”

3.1. Загальний опис системи

Розроблюване програмне забезпечення у вигляді WEB-застосунку із мобільною WEB версією має систематизувати усі процеси у компанії та надавати зручний інструмент для роботи із співпрацівниками у будівельно-інженерних компаніях. Воно має забезпечувати функціонування ієрархічної системи всередині компанії, включаючи надання та контроль виконання завдань, доступ до проектно-кошторисної документації та ДБН, ведення відповідних журналів.

У системі передбачені чотири типів користувачів:

- адміністратор;
- топ-менеджер;
- менеджер;
- звичайний користувач.

Адміністратор

Проводить усі початкові дії, що стосуються інтеграції проекту до програмного забезпечення, реєструє користувачів трьох типів –топ-менеджер, менеджер та звичайний користувач. Має доступ до усіх завдань, що створються всередині актуального проекту, та у інших проектах у даному програмному забезпеченні, де він також зареєстрований у статусі адміністратора. Має доступ до журналу змін, куди внесені усі дії користувачів, зв’язаних із відповідним проектом, щодо завдань чи зміни інформації на власній сторінці.

Менеджер

Має можливість створювати завдання, ставити дату кінцевого терміну виконання. Може назначати завдання користувачам для виконання та відмічати лише тих, хто є серед його підлеглих. Може редагувати завдання, змінюючи текст чи картинки у додатку, продовжуючи або скорочуючи термін виконання, додавати коментарі, редагувати їх (власні коментарі) та видаляти.

Має доступ до журналу пропозицій, може генерувати відповідні звіти для подальших дискусій під час зустрічей.

Топ-менеджер

Має такі ж права як і менеджер, але може ініціалізувати завдання для інших топ-менеджерів його рівня для сумісної роботи (за умови підтвердження даного завдання адміністратором).

Звичайний користувач

Працює із завданнями, може залишати коментарі, редагувати їх (власні коментарі) та видаляти. Також звичайний користувач має доступ до журналу пропозицій – може додавати свої побажання від себе та відмічати інших користувачів.

На рис. 3 наведено діаграму використання системи.

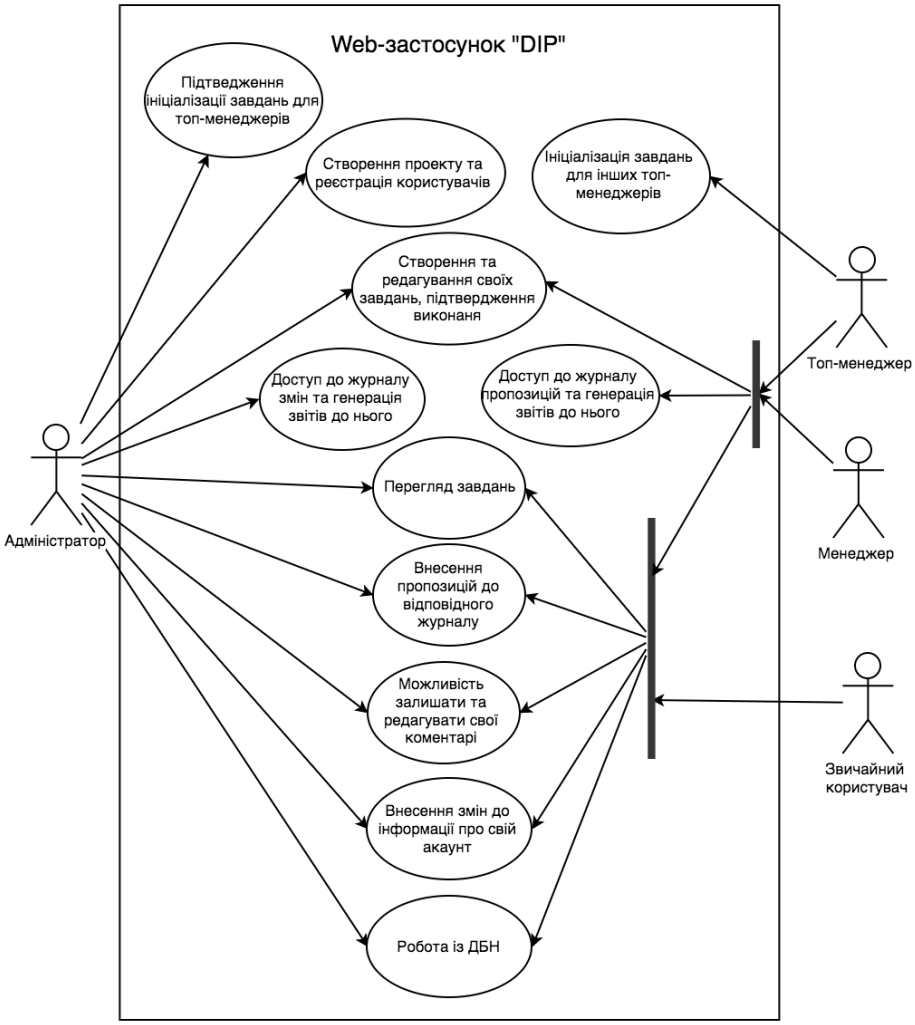


Рис. 3. Діаграма використання

Кожен тип користувача може редагувати інформацію про себе на власній сторінці:

- змінювати фото, ім’я користувача у форматі “@ім’я_користувача”, паролю, електронну пошту;
- додавати інформацію про своє ім’я у месенджері Telegram для активізації розсилки повідомлень про оновлення.

Усі повідомлення про нові завдання, зміни у них, наближення дати кінцевого терміну виконання, та коментарі, де згадується користувач у форматі “@ім’я_користувача” автоматично надсилатимуться користувачу на вказану електронну пошту, та за умови активізації розсилки у Telegram, дублюватимуться туди ж через спеціального Telegram бота.

На рис. 4 наведено UML діаграму роботи застосунку для користувача типу Адміністратор.

На початку роботи із застосунком користувач потрапляє на головну сторінку. Якщо він вже авторизований у системі, він переадресовується на сторінку із завданнями. Якщо користувач неавторизований, він направляється на сторінку авторизації, де автоматично відкривається вікно із полями, які необхідно заповнити для входу у систему – username (ім’я користувача) та password (пароль).

З головної сторінки (автоматично відкрита сторінка із завданнями) користувач може потрапити на такі сторінки відповідно:

- Settings – сторінка профілю користувача, де він може вносити зміни у інформацію про себе та налаштовувати розсилку нових повідомлень;
- Admin – у випадку, якщо користувач типу “Адміністратор”, він може почати роботу із панеллю адміністратора, що включає у собі створення проекту та реєстрацію нових користувачів, роботу із журналом змін та генерацію відповідних звітів;

- Inbox – сторінка із останніми новинами (повідомленнями) для користувача, у разі, якщо він був відмічений у завданні чи він є автором або виконавцем завдання, де відбулися зміни;
- Suggestions – сторінка, де усі типи користувачів можуть вносити пропозиції, лише менеджери та топ-менеджери можуть переглядати внесені пропозиції та генерувати відповідні звіти;
- Rules – сторінка, де розміщені правила ДБН;
- Log Out – дія для виходу із системи, у разі підтвердження намірів виходу із системи, користувач більше не є авторизованим та переадресовується на сторінку авторизації, після чого він може завершити роботу із застосунком чи авторизуватися ще раз і продовжити роботу.

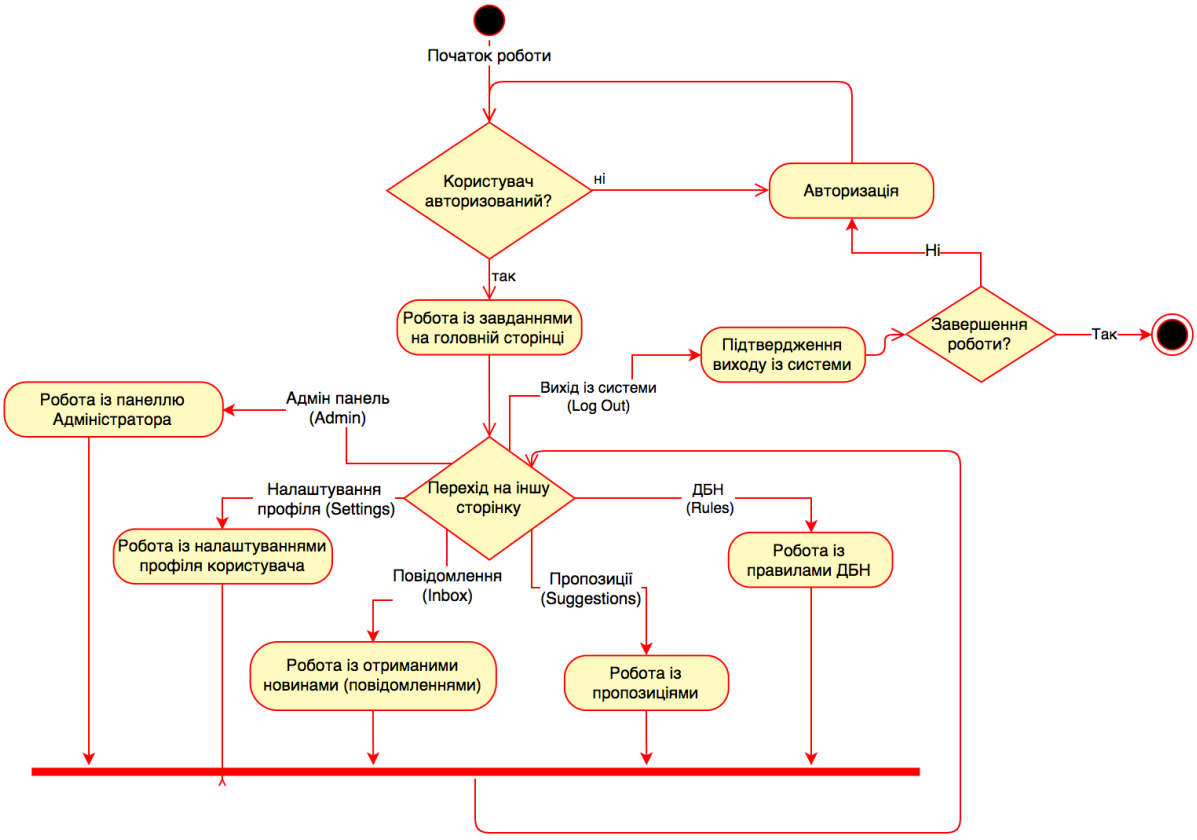


Рис. 4. UML діаграма роботи застосунку для користувача типу Адміністратор

3.2. Архітектура системи

Використання моделі "Model-View-Controller" у Django

Оскільки для розробки було обрано мову програмування Python, було використано відкритий фреймворк на Python для розробки WEB-систем – Django [11].

Архітектура Django подібна до моделі "Model-View-Controller" (MVC), однак має свої особливості. Якщо прямо проектувати модель MVC у Django, тоді вона матиме такий вигляд:

- Model (Модель) надає доступ до даних, оброблюється шаром роботи з базою даних;
- View (Представлення) визначає, які дані отримувати і як їх відображати, оброблюється представленнями та шаблонами;
- Controller (Контролер) обирає представлення у залежності від користувацького запросу, оброблюється самою середою розробки згідно створеної схеми [12].

Так як Controller оброблюється самою середою розробки, то коректніше називати Django MTV-орієнтованим:

- Model (Модель) – робота із даними: як отримати доступ до них, як їх перевірити, використати, як із ними працювати далі так як дані зв'язані між собою;
- Template (Шаблон) – шар представлення даних, який визначає, що і як повинно відображатися на сторінці користувача;
- View (Представлення) – шар бізнес-логіки, який показує, як отримати доступ до моделей і використати відповідний шаблон. Можна його вважати певним "мостом" між моделями та шаблонами [13].

Взаємодія із СКБД

Так як для розробки було обрано мову програмування Python та збереження даних СКБД PostgreSQL, для їх взаємодії було використано

бібліотеку, яка офіційно підтримується Django та сумісна із Django ORM – Psycopg2.

Psycopg2 є найпопулярнішою бібліотекою для організації взаємодії між Python та СКБД PostgreSQL. Вона створена мовою програмування C на базі бібліотеки libpq [14].

Ця бібліотека є драйвером PostgreSQL, сумісним із Python, та знаходиться на стадії активного розвитку. Даний драйвер призначений для багатопоточних застосунків і управляє власним пулом підключень. Головною рисою даного адаптера є те, що при використанні типу даних PostgreSQL, Psycopg2 автоматично перетворює результат та переформатовує його у відповідний тип даних у Python.

Схема бази даних

Для авторизації у застосунку була використана система аутентифікації Django. Оскільки для стандартного класу User (Користувач) вже були задані обов’язкові поля, було розширено клас користувач за допомогою наслідування AbstractUser та додано відповідні поля.

Для реалізації розроблюваного застосунку було створено 14 таблиць, взаємозв’язаних між собою. Із них 6 таблиць є основними:

- User;
- Project;
- Task;
- Comment;
- Suggestion;
- Change.

Інші 8 таблиць є допоміжними та організовують зв’зки у базі даних Many-to-Many та One-to-Many:

- Link_users;
- Task_check;

• Suggestion_followers;

• Task_followers;

• Comment_followers;

• Pic_user;

• Pic_task;

• Pic_comment.

На рис. 5 зображено розроблену схему бази даних.

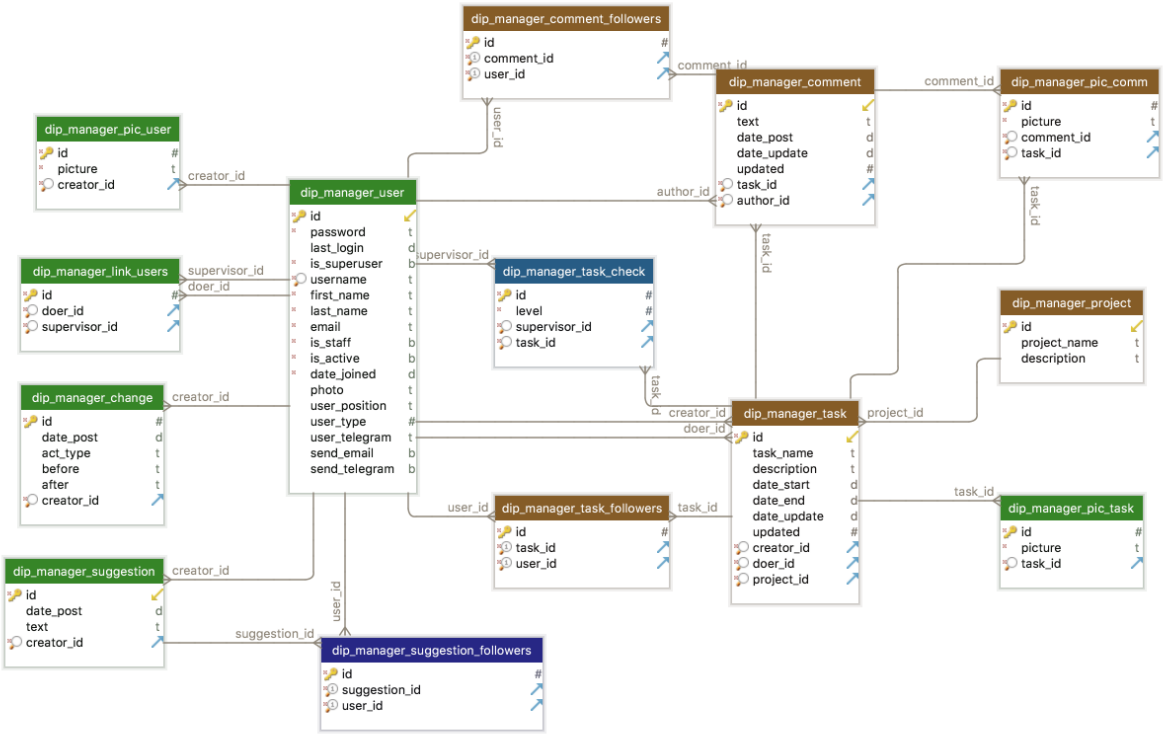


Рис. 5. Схема бази даних системи

User

Таблиця, що містить у собі список усіх користувачів та інформацію про них та відповідну активність. Більш детальна інформація про назви полів на їх типи наведена у табл. 3.

Таблиця 3

Поля таблиці User

Назва поля	*Тип даних*
id	serial AUTOINCREMENT
password	varchar(max_length = 128)
last_login	timestamptz
is_superuser	bool
username	varchar(max_length = 150)
first_name	varchar(max_length = 30)
last_name	varchar(max_length = 150)
email	varchar(max_length = 254)
is_staff	bool
is_active	bool
date_joined	timestamptz
photo	varchar(max_length = 100)
user_position	text
user_type	integer
user_telegram	text

Project

Дана таблиця містить у собі інформацію про проекти, що використовуються у компанії. Більш детальна інформація про назви полів на їх типи наведена у табл. 4.

Таблиця 4

Поля таблиці Project

Назва поля	*Тип даних*
id	serial AUTOINCREMENT
project_name	text
description	text

Task

Таблиця, що містить у собі список усіх завдань, детальну інформацію про них, дату створення, останнього редагування, дату кінцевого виконання завдання, тощо. Більш детальна інформація про назви полів на їх типи наведена у табл. 5.

Таблиця 5

Поля таблиці Task

Назва поля	*Тип даних*
id	serial AUTOINCREMENT
task_name	text
description	text
date_start	timestamptz
date_end	timestamptz
date_update	timestamptz
updated	integer
creator_id	Integer (Foreign Key of User)
doer_id	Integer (Foreign Key of User)
project_id	Integer (Foreign Key of Project)

Comment

У таблиці Comment зберігаються усі коментарі, що були створені у системі, а також інформація про відмічених користувачів і автора. Більш детальна інформація про назви полів на їх типи наведена у табл. 6.

Таблиця 6

Поля таблиці Comment

Назва поля	*Тип даних*
id	serial AUTOINCREMENT
text	text
date_post	timestamptz
date_update	timestamptz
updated	Integer
task_id	Integer (Foreign Key of Task)
creator_id	Integer (Foreign Key of User)

Suggestion

Таблиця Suggestion містить список усіх пропозицій, що були внесені у систему користувачами. Більш детальна інформація про назви полів на їх типи наведена у табл. 7.

Таблиця 7

Поля таблиці Suggestion

Назва поля	*Тип даних*
id	serial AUTOINCREMENT
date_post	timestamptz
text	text
creator_id	Integer (Foreign Key of User)

Change

Таблиця Change зберігає інформацію про будь-які зміни, що відбувалися у системі. Більш детальна інформація про назви полів на їх типи наведена у табл. 8.

Таблиця 8

Поля таблиці Change

Назва поля	*Тип даних*
id	serial AUTOINCREMENT
date_post	timestamptz
act_type	text
before	text
after	text
creator_id	Integer (Foreign Key of User)

Link_users

Дана таблиця є допоміжною та виконує роль зв’язків Many-to-Many, створюючи відповідні пари – керівник та підлеглий. Більш детальна інформація про назви полів на їх типи наведена у табл. 9.

Таблиця 9

Поля таблиці Link_users

Назва поля	*Тип даних*
id	serial AUTOINCREMENT
doer_id	Integer (Foreign Key of User)
supervisor_id	Integer (Foreign Key of User)

Task_check

Дана таблиця є допоміжною та виконує роль зв’язків Many-to-Many, створюючи відповідні пари – керівник та підлеглий. Більш детальна інформація про назви полів на їх типи наведена у табл. 10.

Таблиця 10

Поля таблиці Task_check

Назва поля	*Тип даних*
id	serial AUTOINCREMENT
level	Integer
supervisor_id	Integer (Foreign Key of User)
task_id	Integer (Foreign Key of Task)

Suggestion_followers

Дана таблиця виконує роль Many-to-Many об’єднуючи інформацію про пропозиції та користувачів, що були згадані у них. Більш детальна інформація про назви полів на їх типи наведена у табл. 11.

Таблиця 11

Поля таблиці Suggestion_followers

Назва поля	*Тип даних*
id	serial AUTOINCREMENT
suggestion_id	Integer (Foreign Key of Suggestion)
user_id	Integer (Foreign Key of User)

Task_followers

Таблиця Task_followers виконує роль утворення зв’язків Many-to-Many між моделями User (Користувач) та Task (Завдання). Більш детальна інформація про назви полів на їх типи наведена у табл. 12.

Таблиця 12

Поля таблиці Task_followers

Назва поля	*Тип даних*
id	serial AUTOINCREMENT
task_id	Integer (Foreign Key of Task)
user_id	Integer (Foreign Key of User)

Comment_followers

Допоміжна таблиця для зв'язків Many-to-Many між моделями User (Користувач) та Comment (Коментар). Більш детальна інформація про назви полів на їх типи наведена у табл. 13.

Таблиця 13

Поля таблиці Comment_followers

Назва поля	*Тип даних*
id	serial AUTOINCREMENT
comment_id	Integer (Foreign Key of Comment)
user_id	Integer (Foreign Key of User)

Pic_user

Допоміжна таблиця, що допомагає зберігати картинки користувача. Більш детальна інформація про назви полів на їх типи наведена у табл. 14.

Таблиця 14

Поля таблиці Pic_user

Назва поля	*Тип даних*
id	serial AUTOINCREMENT
picture	varchar(max_length = 100)
creator_id	Integer (Foreign Key of User)

Pic_task

Таблиця Pic_task реалізує зв’язок One-to-Many та зберігає декілька картинок у завданні. Більш детальна інформація про назви полів на їх типи наведена у табл. 15.

Таблиця 15

Поля таблиці Pic_task

Назва поля	*Тип даних*
id	serial AUTOINCREMENT
picture	varchar(max_length = 100)
task_id	Integer (Foreign Key of Task)

Pic_comment

Таблиця Pic_task є допоміжною та реалізує зв’язок One-to-Many і допомагає зберігати декілька картинок у коментарі до завдання. Більш детальна інформація про назви полів на їх типи наведена у табл. 16.

Таблиця 16

Поля таблиці Pic_comment

Назва поля	*Тип даних*
id	serial AUTOINCREMENT
picture	varchar(max_length = 100)
comment_id	Integer (Foreign Key of Comment)

3.3. Особливості реалізації

Так як розробка програмного забезпечення велася у відкритому фреймворку Django, у архітектурі якого можна провести паралель із моделлю MVC – MTV особливості реалізації можна розділити на три основні блоки:

- Model;

- Template;
- View.

Model відповідає за повний цикл роботи із даними на стороні сервера. У даному програмному забезпеченні були створені такі класи, що відповідають основним документам у базі даних:

- User – включає в собі:
  - унікальний ідентифікатор;
  - ім’я працівника;
  - прізвище працівника;
  - фото;
  - ім’я користувача у форматі “@ім’я_користувача”;
  - зашифрований пароль;
  - тип користувача;
  - електронну пошту;
  - посада;
  - чи є активним на даний момент;
  - дата приєднання до системи;
  - чи відноситься до технічної команди;
  - чи є суперкористувачем з усіма правами;
  - може включати ім’я користувача у месенджері Telegram.
- Project – включає в собі:
  - унікальний ідентифікатор;
  - назву проекту;
  - опис проекту.
- Task – включає в собі:
  - унікальний ідентифікатор;
  - назву завдання;
  - текст із описом завдання;
  - дату створення даного завдання;

- дату кінцевого терміну виконання даного завдання;
- ForeignKey користувача (клас User), що створив це завдання;
- ForeignKey користувача (клас User), якому назначене це завдання особисто;
- дату останнього оновлення завдання;
- помітку про внесення оновлень у завдання;
- ForeignKey проекту (клас Project), до якого відноситься це завдання.

- Comment – включає в собі:
  - унікальний індентифікатор;
  - дату створення даного коментаря;
  - ForeignKey користувача (клас User), що створив цей коментар;
  - сам текст коментаря;
  - відмітку редагування (у випадку, коли коментар редагується перший раз у нього з’являється така відмітка);
  - дату останнього редагування;
  - ForeignKey завдання (клас Task), до якого відноситься цей коментар.
- Change – включає у собі:
  - унікальний ідентифікатор;
  - дату створення даної зміни;
  - ForeignKey користувача (клас User), чия дія записується;
  - тип дії;
  - текст із інформацією про стан “до” ;
  - текст із інформацією про стан “після”.
- Suggestion – включає в собі:
  - унікальний ідентифактор;
  - дату створення даної пропозиції;
  - текст самої пропозиції;

- ForeignKey користувача (клас User), що створив дану пропозицію.

Також є окремий модуль розсилки повідомлень користувачам по електронній пошті та Telegram бот для дублювання.

Template – це шар представлення даних, який визначає, що і як повинно відображатися на сторінці користувача. Для цього були створені такі сторінки у форматі HTML:

- Index – головна сторінка застосунку. Якщо на неї перейшов користувач, що вже знаходиться у системі, тоді вона переадресовує його на сторінку Tasks відповідно до його акаунту. Якщо на головну сторінку зайшов новий користувач, він може зайти у систему за допомогою ім’я користувача (“@ім’я_користувача”) та паролю у автоматично відкритому вікні входу у систему.
- Tasks – на цю сторінку може перейти лише авторизований користувач. Зліва розташована панель швидкого доступу до різних розділів відповідно до типу користувача:
  - Tasks (активна сторінка);
  - Inbox;
  - Settings;
  - Suggestions;
  - Admin;
  - Rules.
  - На верхній панелі вказані:
    - ім’я користувача;
    - фото користувача у системі.

    При натисненні на фото користувача з’являється випадаючий список, де користувач може вийти із системи (лише після підтвердження у додатковому вікні про бажання вийти із системи), або перейти на сторінку свого акаунту Settings.

У основному полі сторінки розміщені завдання для користувача, відсортовані у порядку від найближчих до завершення кінцевого терміну виконання завдання, до найдальших. Користувач може відкрити завдання, натиснувши на нього, переглянути, додати коментарі та відмітити (за бажанням) користувачів у форматі “@ім’я_користувача”.

Користувач типу менеджер може створювати завдання, назначати користувачів для його виконання, ставити дату кінцевого терміну виконання. Може редагувати завдання, змінюючи текст чи картинки у додатку, продовжуючи або скорочуючи термін виконання, додавати коментарі, редагувати їх (власні коментарі) та видаляти.

- Inbox – сторінка із повідомленнями про створення завдання самим користувачем, чи на які він був назначений або відмічений у форматі “@ім’я_користувача” в тексті самого завдання чи у коментарях, нагадування про приближення дати кінцевого терміну виконання завдання.
- Settings – сторінка користувача, де він може змінити:
  - фото;
  - ім’я користувача у форматі “@ім’я_користувача”;
  - пароль;
  - електронну пошту.

  Також кожен користувач може активізувати розсилку із новинами через Telegram бота, ввівши у відповідний рядок своє користувацьке ім’я у месенджері Telegram.

- Suggestions – для звичайного користувача це сторінка із однією кнопкою “Подати пропозицію”, натиснувши на яку користувач може ввести текст пропозиції, відмітити (за бажанням) користувачів у форматі “@ім’я_користувача”, після чого відправити свою пропозицію. Для користувачів типу менеджер ця сторінка відображає

пропозиції, що надходили від працівників, є можливість надіслати свою пропозицію та автоматично згенерувати звіт по окремим критеріям (дата, користувачі).

- Admin – сторінка, доступна лише для користувача типу адміністратор. Тут він може:
  - створювати новий проект;
  - додавати працівників (вводячи повне ім’я та посаду), автоматично генеруючи для них пароль та ім’я користувача для першого входу у систему;
  - запускати автоматизовану розсилку початкових завдань;
  - працювати із журналом змін та автоматично генерувати звіт по окремим критеріям (користувач, дата, тип дії).
- Rules – сторінка, доступна для усіх користувачів, являє собою пошукову систему по ДБН.

Також додатково використовувались бібліотеки зі стилями та анімацією, що знаходяться у відкритому доступі – Bootstrap, sb-admin2.

View – шар бізнес-логіки, який показує, як отримати доступ до моделей і використати відповідний шаблон. У розроблюваному програмному забезпеченні View можна розділити на дві частини:

- API – для виконання додаткових внутрішніх запросів;
- основна – для взаємодії системи із користувачем.

3.4. Висновки до розділу

У даному розділі було проведено загальний опис розроблюваного програмного забезпечення у вигляді WEB-додатку із мобільною WEB-версією. Також було описано та розподілено права трьох типів користувачів – адміністратор, менеджер, звичайний користувач.

При описі архітектури системи було проведено аналогію із загальновживаною моделлю MVC (“Model-View-Controller”)

та альтернативною: що використовується у Django – MTP (“Model-Template-Controller”).

Для підтримання взаємодії між функціонуючою системою та СКБД було обрано бібліотеку, яка офіційно підтримується Django та сумісна із Django ORM – Psycopg2.

4. АНАЛІЗ РОЗРОБЛЕНОГО WEB-ЗАСТОСУНКУ “DIP”

4.1. Аналіз реалізованого WEB-застосунку

У результаті розроблення WEB-застосунку “DIP” із мобільною WEB-версією було створено ПЗ, призначене для систематизації усіх процесів у компанії та надання зручного інструменту для роботи співпрацівникам будівельно-інженерних компаній.

Відповідно до сформованих функціональних вимог до системи (див. підрозділ 1.4. “Загальні вимоги до системи”) було розроблено відповідні рішення:

- клас Користувач використовує вбудовану Django Admin авторизацію для підтримання рівня безпеки під час авторизації та відслідковування сесій. Він є розширеним класом AbstractUser із додатковими полями для інтегрування у розроблювану систему. Різні рівні доступу користувачів “Адміністратор”, “Топ-менеджер”, “Менеджер”, “Звичайний користувач” забезпечуються відповідними полями у класі користувача, де вказується їх рівень та посада. Також було створено додатковий клас, який допомагає відтворювати пари “керівник – підлеглий” для формування відповідної ієрархічної системи.
- Система передачі та контролю виконання завдань реалізована за допомогою класів Завдання та Коментар. Кожне завдання має інформацію про користувачів, із якими воно пов’язане (вони відмічені у завданні чи є автором або виконавцем). Користувачі типу “Адміністратор”, “Топ-менеджер” та “Менеджер” можуть створювати нові завдання, редагувати їх, назначати їх відповідно користувачам, які є їх підлеглими, чи відмічати їх у форматі “@ім’я_користувача”. Лише користувач типу “Топ-менеджер” може ініціювати завдання для інших користувачів типу “Топ-менеджер” для спільної роботи над ним за умови підтвердження ініціації адміністратором. Користувач

будь-якого типу може додавати коментарі до завдань, редагувати та видаляти власні коментарі.

- У випадку, коли Користувач 1 – автор завдання, Користувач 2 – виконавець завдання і при цьому рівні цих користувачів відрізняються більше ніж на один, тобто виконавець не є прямим підлеглим автора (але, наприклад, є підлеглим менеджера, яким управляє Користувач 1), система автоматично генерує завдання для усіх користувачів типу “Менеджер” у відповідній ієрархічній гілці від Користувача 1 до Користувача 2. Наприклад, Користувач 1 (виконавець) – користувач типу “Звичайний користувач”, його рівень – 1. Користувач 2 (автор) – користувач типу “Топ-менеджер”, його рівень – 3. Користувач 1 є підлеглим Користувача 2, але непрямим, тому що Користувач 1 є прямим підлеглим Користувача 3 – користувач типу “Менеджер”, рівень – 2, який відповідно є прямим підлеглим Користувача 1. У даному випадку при створенні завдання Користувачем 1 для Користувача 2, Користувач 3 автоматично отримає завдання для Користувача 2. Після виконання завдання Користувачем 2, спочатку підтвердити його виконання матиме Користувач 3 і лише потім Користувач 1. Після підтвердження виконання завдання користувачем автором – воно зникає у всіх користувачів, що були задіяні у ньому.

- Ведення журналу пропозицій реалізовано таким чином, що користувачі типу “Звичайний користувач” можуть лише додавати свої пропозиції. Користувачі типу “Менеджер” і “Топ-менеджер” можуть працювати із журналом пропозицій та автоматично генерувати відповідні звіти по критеріям часу та користувачів, що були відмічені.
- Ведення журналу змін виконується системою автоматично при внесення будь-яких змін у систему користувачами, що стосуються завдань, проекту чи особистої інформації, а також оновлення ДБН.

- Робота із налаштуванням власного профіля користувача ведеться через відповідну сторінку, яка доступна усім типам користувачів. Вони можуть змінювати фото, ім’я користувача у форматі “@ім’я_користувача”, паролю, електронну пошту та додавати інформацію про своє ім’я у месенджері Telegram та активізувати Telegram бота для розсилки повідомлень про оновлення. Він надсилатиме повідомлення у форматі “Для вас з’явилося нове завдання” без деталей завдання (задля запобігання порушення вимог до безпеки системи). Також вони можуть деактивізувати розсилку повідомлень про оновлення через пошту.

Розроблене програмне забезпечення для працівників інженерно-будівельних компаній відповідає таким нефункціональним вимогам:

- воно представлене у вигляді WEB-застосунку із мобільною WEB-версією;
- розроблено зручний інтерфейс та протестовано реальними користувачами.

4.2. Тестування WEB-застосунку

Тестування є одним із елементів життєвого циклу продукту та перш за все спрямоване на перевірку коректності роботи системи, пошук помилок, можливих недоліків та їх усунення. Також даний процес допомагає відкорегувати графічний інтерфейс і покращити функціональність даного програмного забезпечення у вигляді WEB-застосунку із мобільною WEB-версією за умови надання на тестування реальним потенційним користувачам на різних гаджетах.

Тестування розроблюваного програмного забезпечення проводилося на кожній ітерації створення системи та є невід’ємною частиною процесу створення закінченого програмного продукту. В ході виконання дипломного

проекту та розроблення WEB-застосунку воно виконувалось після кожного етапу реалізації певних елементів системи.

Як один зі способів тестування було прийнято ручне тестування для перевірки захисту від SQL-ін’єкцій та перевірку введених даних, що включало введення некоректних даних, пустих даних чи даних занадто великого об’єму.

За результатами цього тестування до розроблюваного програмного забезпечення у вигляді WEB-застосунку із мобільною WEB-версією були додані повідомлення на клієнтській частині про відповідні помилки, які виникають при вводі некоректних даних.

Також було проведено тестування потенційними реальними користувачами щодо зручності використання запропонованого графічного інтерфейсу. Усі відгуки та зауваження були ретельно опрацьовані та проаналізовані. Відповідно до них було внесено такі зміни у графічний інтерфейс та функціональність:

- заміна кольорів головної теми на більш нейтральні;
- перенесення кнопки виходу із системи на верхню панель замість лівої бокової панелі;
- додання можливості редагувати фото;
- забезпечення можливості надсилання повідомлень про новини та нагадування до месенджеру Telegram кожному користувачу окремо через спеціально створеного Telegram бота;
- створення та підтримка журналу пропозицій із відповідною автоматизованою генерацією звітів відповідно до критеріїв –період часу та відмічені користувачі.

Отже, тестування розроблюваного програмного забезпечення проводилося на кожній ітерації створення системи у вигляді ручного тестування та наприкінці тестування реальними потенційними користувачами. Було враховано усі недоліки, що виникли, та побажання користувачів щодо

функціональності та зручності графічного інтерфейсу і реалізовано їх у розроблюваній системі.

4.3. Порівняння розробки із існуючими аналогами

У першому розділі дипломного проекту було розглянуто декілька аналогів, що використовуються у будівельно-інженерних компаніях. Усі аналоги були вивчені із метою виявлення їх переваг та недоліків. Розглянуті продукти лише частково мають спільні функції із розроблюваним програмним забезпеченням, оскільки існуючі комплексні аналоги розробляються виключно у індивідуальному порядку для великих будівельно-інженерних компаній і коштують дуже дорого і не публікуються у відкритому доступі.

Головними критеріями оцінки були:

- універсальність – розроблений WEB-застосунок надає зручний інструмент для управління працівниками компаніям будь-якого розміру із різноплановими проектами;
- простота використання – після тестування реальними потенційними користувачами система була визнана більш зручною у використанні, ніж наведені для прикладу існуючі аналоги;
- зручність інтерфейсу – дана система проходила тестування реальними потенційними користувачами, після чого були скореговані усі виявлені недоліки;
- кросплатформність – розроблене програмне забезпечення представлене у вигляді WEB-застосунку із мобільною WEB-версією, що робить використання на будь-якому сучасному гаджеті можливим, оскільки усі представлені у першому розділі аналоги надаються у вигляді програмного забезпечення, що треба завантажувати та встановлювати на комп’ютер і, крім того, лише для ОС Windows;

- доступність – розроблене програмне забезпечення є доступним для компаній будь-якого розміру, оскільки є універсальним та не є індивідуальною розробкою для окремої великої компанії.

Отже, порівнявши аналоги, що представлені на ринку будівельно-інженерної галузі на даний момент, можна дійти до висновку, що розроблене програмне забезпечення у вигляді WEB-додатку із мобільною WEB-версією може конкурувати на гідному рівні із існуючими продуктами.

4.4. Рекомендації щодо подальшого вдосконалення

Вибір функцій розробленого програмного забезпечення у вигляді WEB-застосунку є достатньо широким з огляду на зручність використання працівниками та задовольняє багато потреб водночас, однак дана система має бути універсальною, як згадувалося вище та комплексною, тому її необхідно весь час вдосконалювати.

Було проведено опитування серед цільової аудиторії серед топ-менеджменту будівельно-інженерних компаній, згідно до якого існує ряд додаткових функцій, що значно вдосконалили б розроблюване програмне забезпечення:

- автоматизований кошторис;
- робота із кресленнями;
- проекція креслень на стіну;
- внесення змін до проектно-кошторисної документації та ведення відповідного журналу внесення змін із автоматичною генерацією звітів;
- введення системи електронного підпису задля пришвидшення прийняття рішень, знаходячись прямо на об’єкті без необхідності поїздки до офісу для підтвердження;
- помітки, зручний пошук по проектно-кошторисній документації;

- месенджер для працівників інженерно-будівельної компанії з можливістю прикріплення до повідомлень завдань у системі із відповідними вказівками;
- можливість прикріплення елементів проектно-кошторисної документації та ДБН до завдань та коментарів у системі;
- вбудований “рівень” та рулетка для виміру розмірів об’ємних фігур у просторі.

Якщо даний проект продовжуватиме активно розвиватися, у майбутньому можлива перспектива повної автоматизації процесу будівництва за допомогою спеціальних машин, більш масштабних аналогів 3D принтерів.

4.5. Висновки до розділу

У даному розділі було проведено аналіз реалізованого програмного забезпечення у вигляді WEB-застосунку із мобільною WEB-версією. Було проведено аналогію із поставленими функціональними і нефункціональними вимогами та відповідно описано методи та рішення, що були прийняті, і яким способом вони розроблені.

На кожній ітерації створення системи проводилося ручне тестування та вносилися відповідні правки. Наприкінці було проведено тестування реальними потенційними користувачами. Було враховано усі недоліки, що виникли, та побажання користувачів щодо функціональності та зручності графічного інтерфейсу і реалізовано їх у розроблюваній системі.

При порівнянні із аналогами, що представлені на ринку будівельно-інженерної галузі на даний момент, було зроблено висновок, що розроблене програмне забезпечення у вигляді WEB-додатку із мобільною WEB-версією може конкурувати на гідному рівні із існуючими продуктами.

Також було проведено опитування цільової аудиторії серед топ-менеджменту будівельно-інженерних компаній, згідно до якого існує ряд додаткових функцій, що значно вдосконалили б розроблюване програмне

забезпечення, серед яких автоматичне формування кошторису, робота із кресленнями, можливість внесення змін до проектно-кошторисної документації із автоматичною генерацією змін, введення електронного підпису, тощо. Відповідно було сформовано перелік функцій, що є рекомендаціями для подальшого вдосконалення застосунку.

ВИСНОВКИ

Метою даного дипломного проекту було створення універсального та доступного програмного забезпечення, що призначене для систематизації усіх процесів у компанії та надання зручного інструменту для роботи співпрацівникам будівельно-інженерних компаній.

Було проведено аналіз існуючих проблем у будівельно-інженерній галузі, розглянуто представлені на ринку аналоги та відповідно сформовано вимоги до розроблюваної системи у вигляді функціональних, нефункціональних вимог та вимог до безпеки.

Після порівняння засобів реалізації було обрано створення програмного забезпечення у вигляді WEB-застосунку з мобільною WEB-версією мовою програмування Python за допомогою фреймворку Django із використанням СКБД PostgreSQL.

Було створено програмне забезпечення “DIP” для інженерно-будівельних компаній у вигляді WEB-застосунку з мобільною WEB-версією, що:

- забезпечує функціонування ієрархічної системи всередині кожної окремої компанії, що використовує даний застосунок, а саме:
  - можливість керування інформацією про працівників;
  - надання вказівок конкретним працівникам або їх групам та контроль їх виконання;
  - зворотній зв’язок із керівничим складом компанії (журнал пропозицій);
- веде відповідні журнали: із зворотнім зв’язком та проблемами, що виникають (журнал пропозицій) та про будь-які зміни у системі (журнал змін);
- надає доступ до ДБН;
- підтримує відповідний рівень безпеки системи та її швидкодію;
- має зручний інтерфейс.

Розробка програмного забезпечення виконана у повному обсязі та відповідає поставленим вимогам. Тестування системи виконано у відповідності до затвердженої програми та методики тестування.

Програмне забезпечення “DIP” є універсальним та доступним інструментом для систематизації та контролю усіх процесів у компанії.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

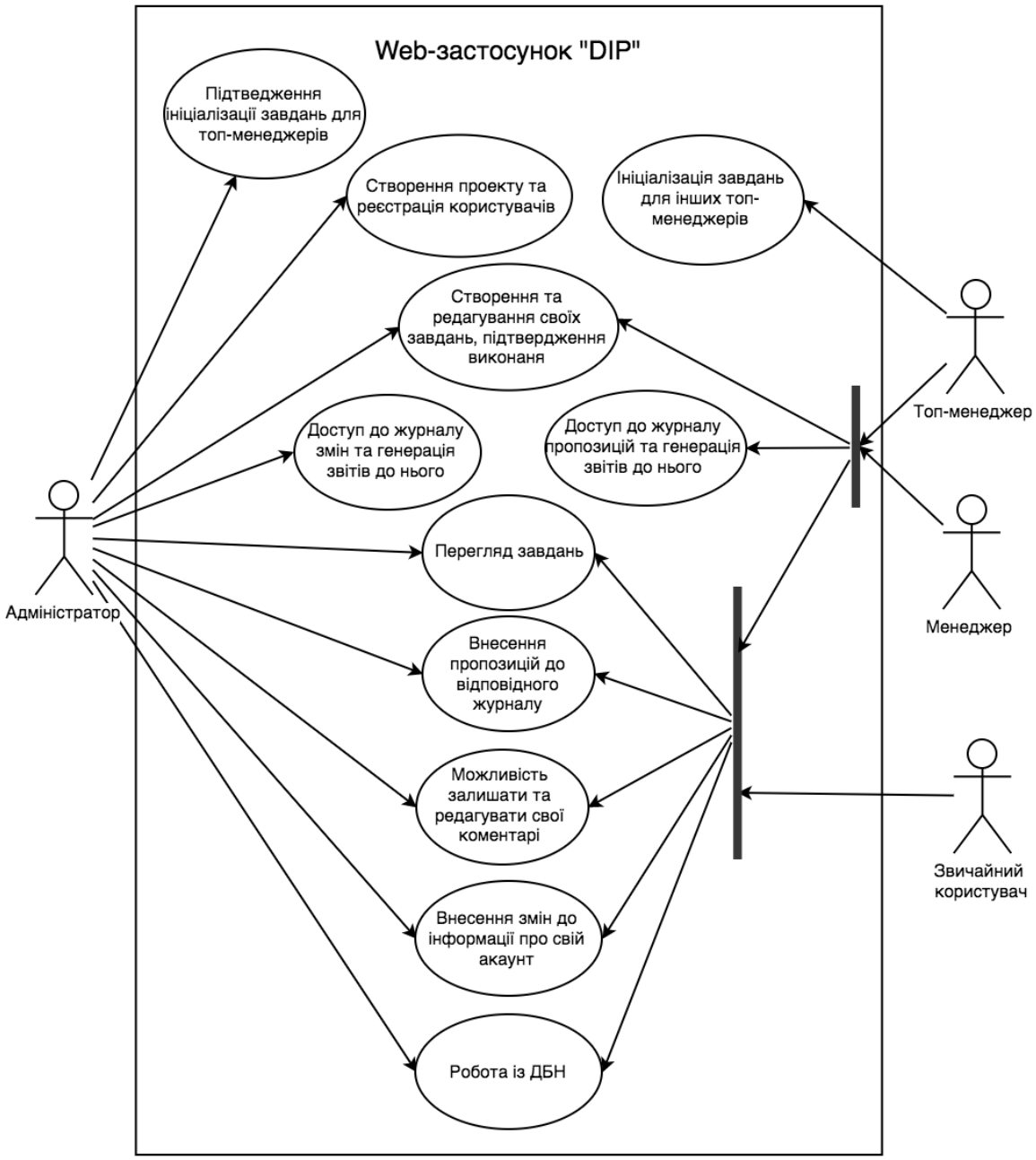
1. Научно-производственная фирма «АВК Созидатель» : [Електронний ресурс]. Режим доступу: https://avk5.com.ua/
2. ПК «Гранд-Смета» : [Електронний ресурс]. Режим доступу: https://www.grandsmeta.ru/product/pk-grand-smeta
3. Python documentation : [Електронний ресурс]. Режим доступу: https://www.python.org/doc/
4. Краткая характеристика трёх языков программирования : [Електронний ресурс]. Режим доступу: http://www.internet-technologies.ru/articles/article_1991.html
5. Web Frameworks for Python : [Електронний ресурс]. Режим доступу: https://wiki.python.org/moin/WebFrameworks
6. Django framework : [Електронний ресурс]. Режим доступу: https://www.djangoproject.com/
7. Welcome to MySQL : [Електронний ресурс]. Режим доступу: https://www.mysql.com/
8. Welcome to PostgreSQL : [Електронний ресурс]. Режим доступу: http://www.postgresql.org/
9. Microsoft SQL server 2019 : [Електронний ресурс]. Режим доступу: https://www.microsoft.com/ru-ru/sql-server/sql-server-2019
10. Welcome to MongoEngine : [Електронний ресурс]. Режим доступу: http://mongoengine.org/
11. Django documentation : [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/1.8/
12. Django book: [Електронний ресурс]. Режим доступу: https://djbook.ru/ch05s02.html
13. Understanding of MVC pattern in Django : [Електронний ресурс]. Режим доступу: https://medium.com/shecodeafrica/understanding-the-mvc-pattern-in-django-edda05b9f43f

14. Psycopg2 documentation : [Електронний ресурс]. Режим доступу: http://initd.org/psycopg/docs/

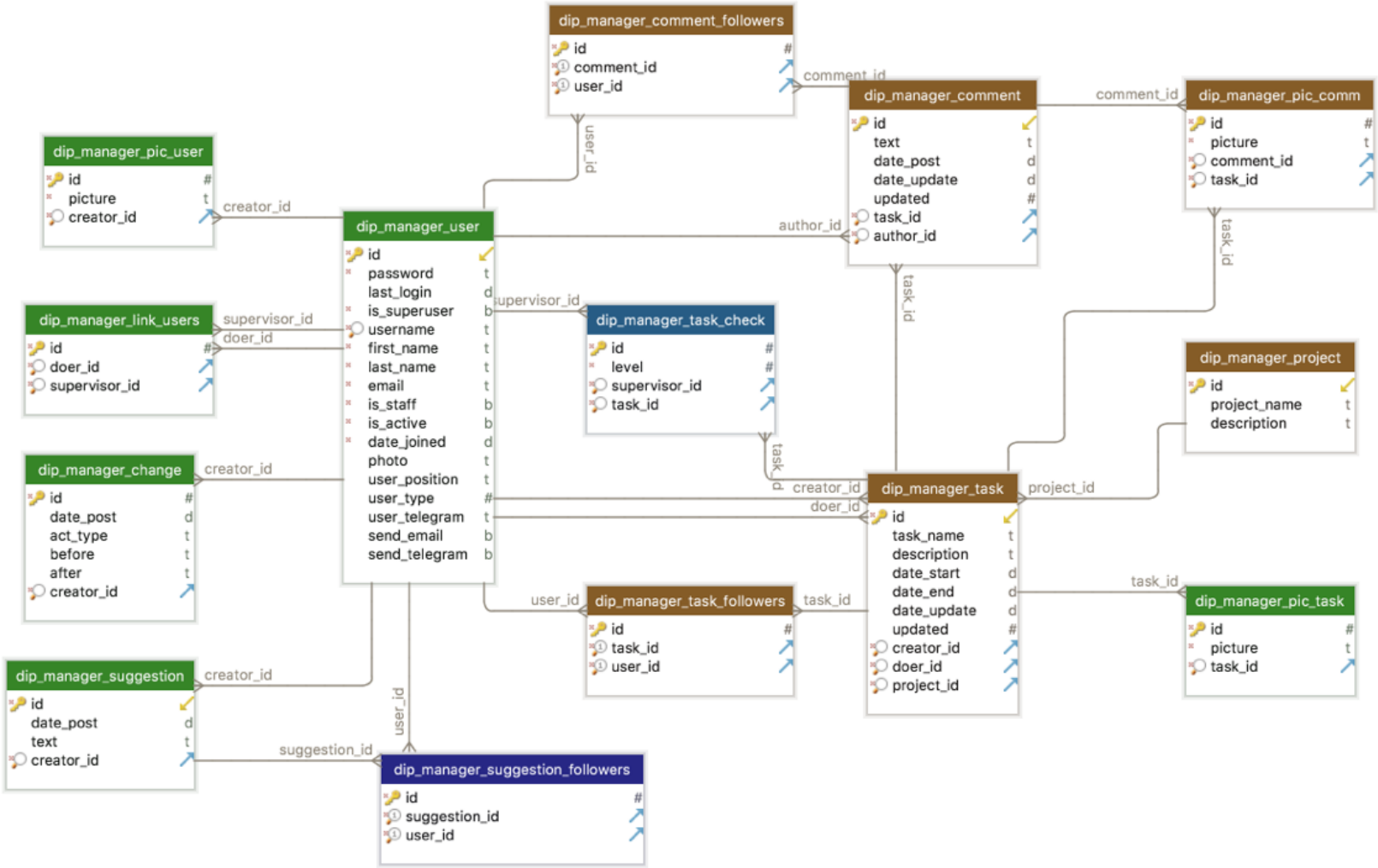
ДОДАТКИ

Додаток 1

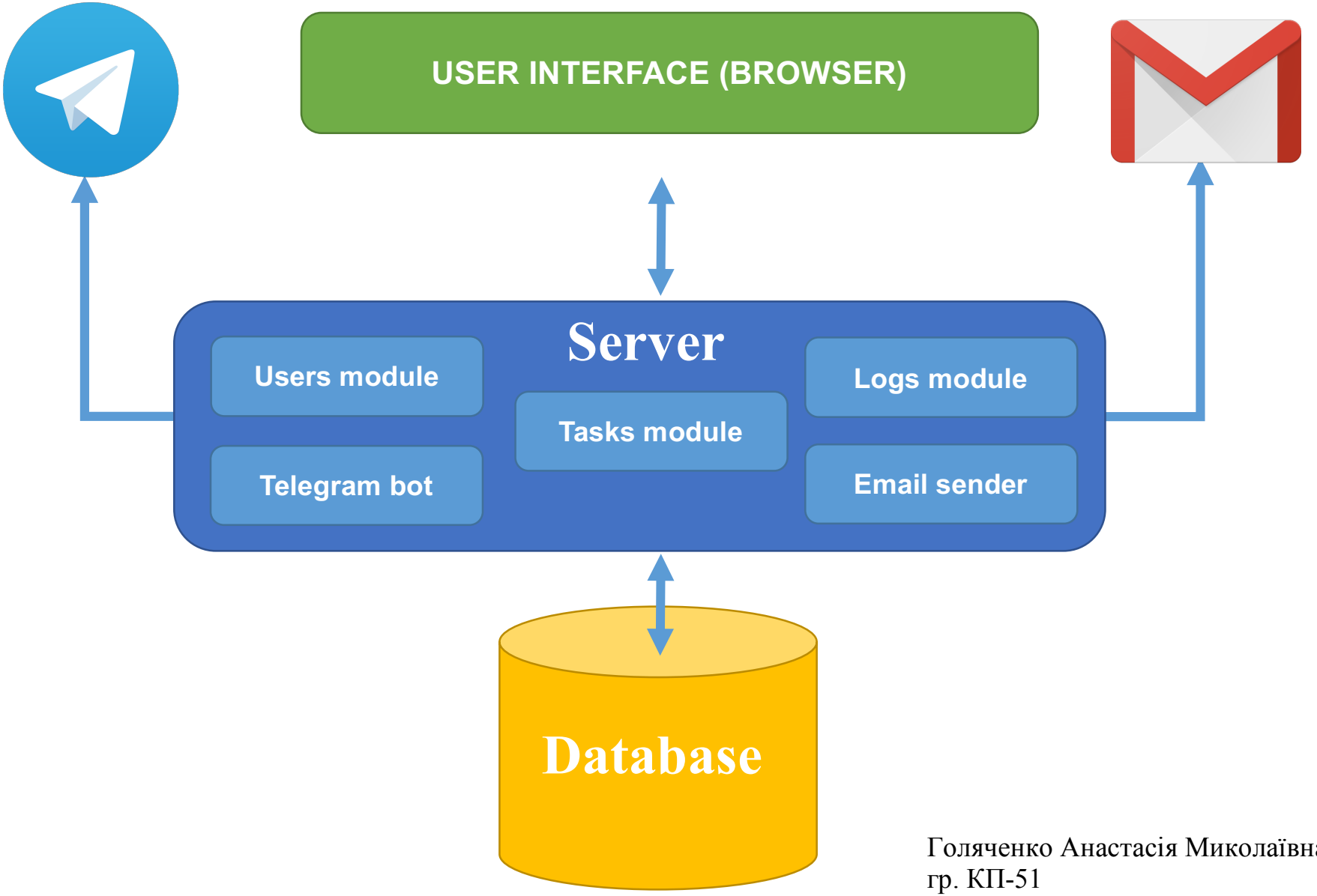
Копії графічних матеріалів



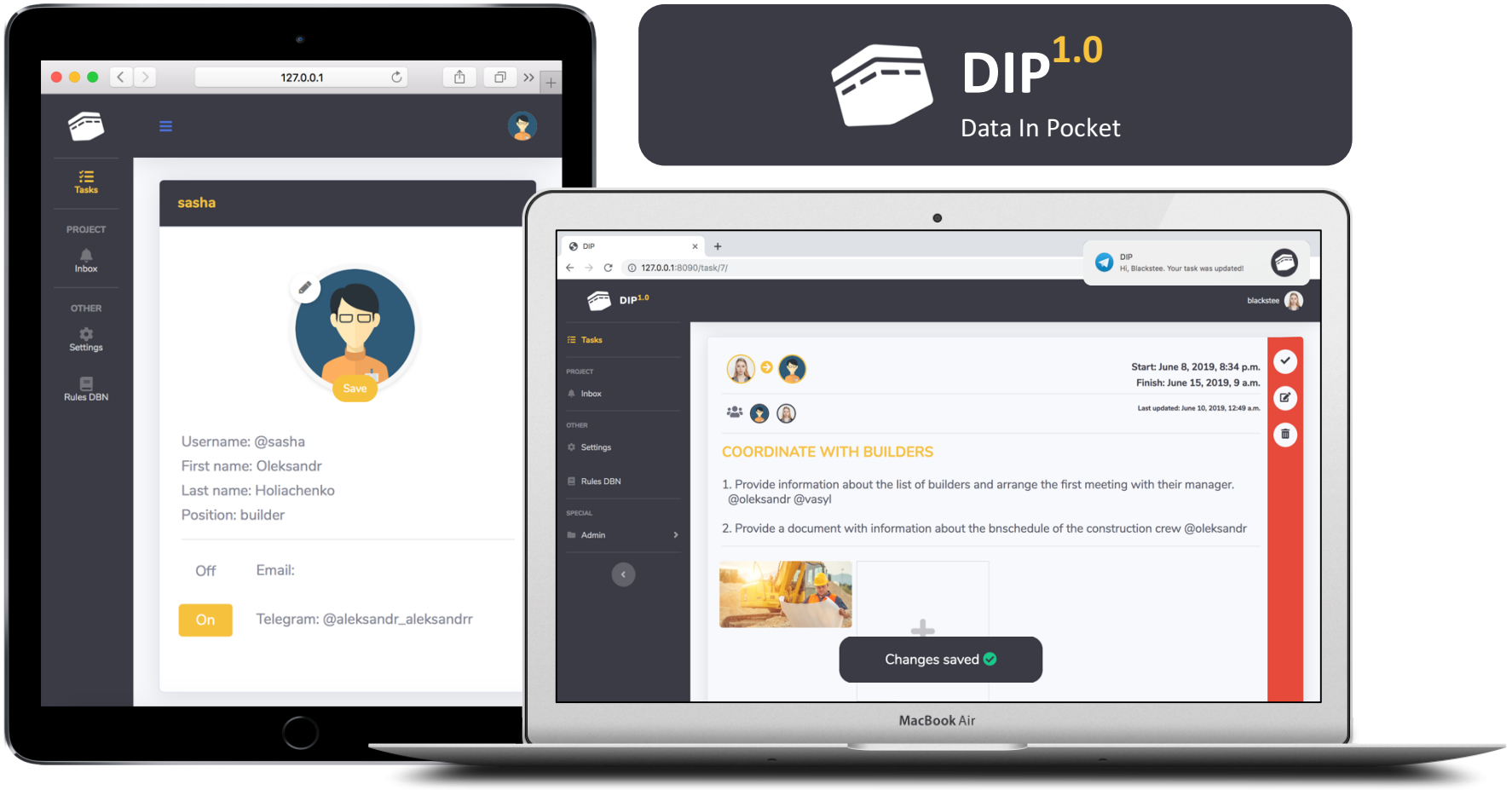
ДП.045440-06-99. Автоматизована система управління постановкою та контролем виконання задач для будівельно-інженерних компаній. Діяльність користувача у системі. Діаграма використання



ДП.045440-07-99. Автоматизована система управління постановкою та контролем виконання задач для будівельно-інженерних компаній. Архітектура системи. Схема бази даних



Голяченко Анастасія Миколаївна,
гр. КП-51



Голяченко Анастасія Миколаївна,
гр. КП-51

Додаток 2

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

АВТОМАТИЗОВАНА СИСТЕМА УПРАВЛІННЯ ПОСТАНОВКОЮ ТА КОНТРОЛЕМ ВИКОНАННЯ ЗАДАЧ ДЛЯ БУДІВЕЛЬНО-ІНЖЕНЕРНИХ КОМПАНІЙ

Виконала: Голяченко Анастасія Миколаївна

Науковий керівник: Ст. викладач кафедри ПЗКС, к.т.н., Люшенко Л.А.

Київ – 2019

ПОСТАНОВКА ЗАДАЧІ

Мета проекту: розробити WEB-застосунк з мобільною WEB-версією для постановки і контролю виконання завдань, та оптимізації управлінських процесів будівельно-інженерної компанії.

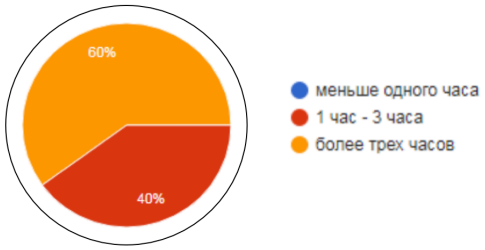
Завдання:

1. Проаналізувати існуючі на ринку аналоги та сформувати вимоги до ПЗ
2. Обрати та обґрунтувати засоби реалізації ПЗ
3. Розробити ПЗ згідно поставлених вимог
4. Протестувати систему, виправити виявлені недоліки
5. Проаналізувати розроблену систему та сформулювати рекомендації до подальшого вдосконалення
6. Сформувати висновки

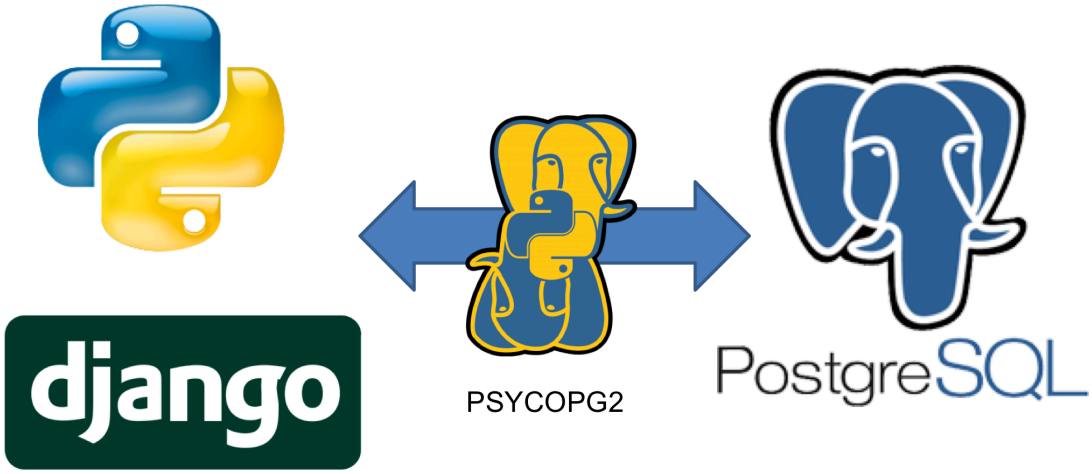
АКТУАЛЬНІСТЬ

- У більшості будівельно-інженерних компаній не використовується спеціалізоване програмне забезпечення для автоматизації управління процесами будівництва
- Це призводить до неупорядкованості інформації, плутанини, неякісно чи неправильно виконаної роботи, зайвих витрат часу і відповідно грошей.

Результати опитування серед топ-менеджерів



Засоби реалізації



Архітектура системи

Django MTV (аналог MVC):
- Model
- View
- Template

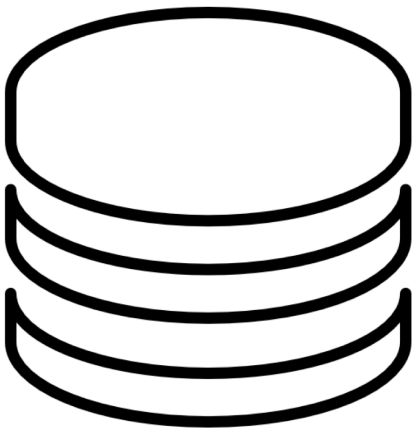
База даних: PostgreSQL

Використання додаткових сервісів:
Telegram та Gmail

Архітектура бази даних

Головні таблиці:
- Користувач
- Проект
- Завдання
- Коментар
- Пропозиція
- Зміна

- 8 допоміжних таблиць



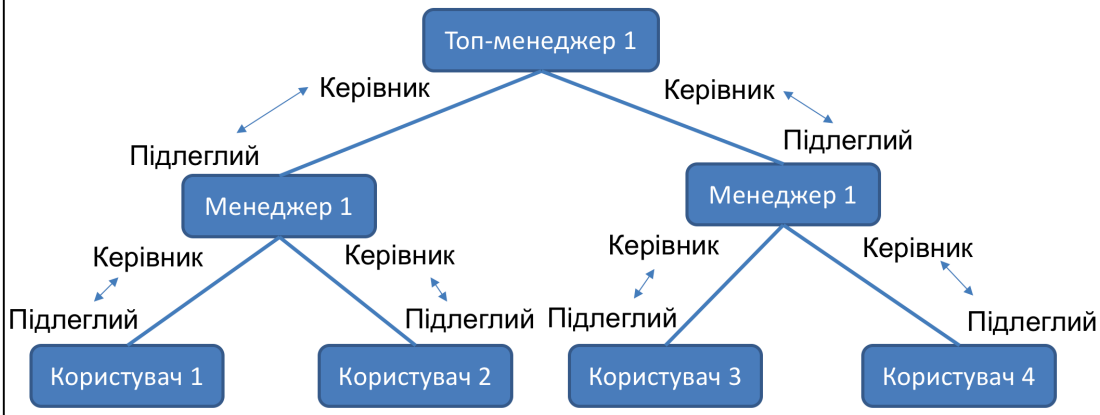
Діаграма використання

Чотири типи користувачів із різними рівнями прав доступу:

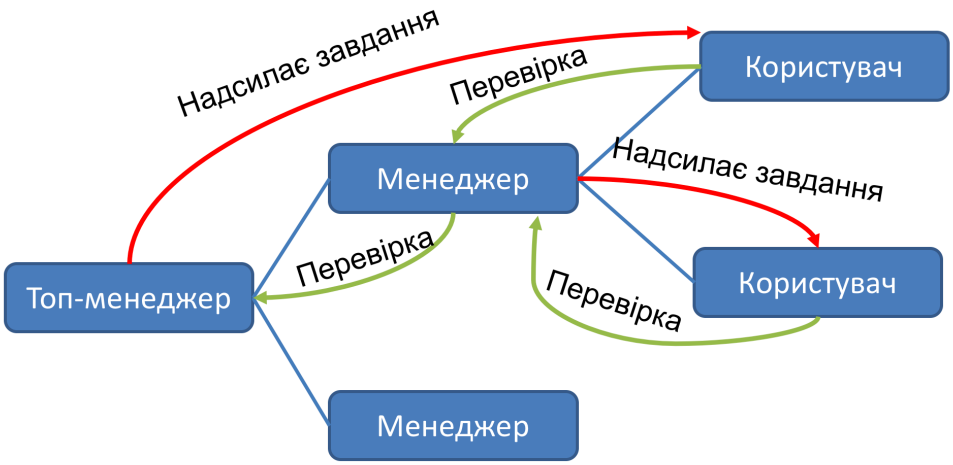
- Адміністратор
- Топ-менеджер
- Менеджер
- Звичайний користувач

Алгоритм генерації ієрархічної структури

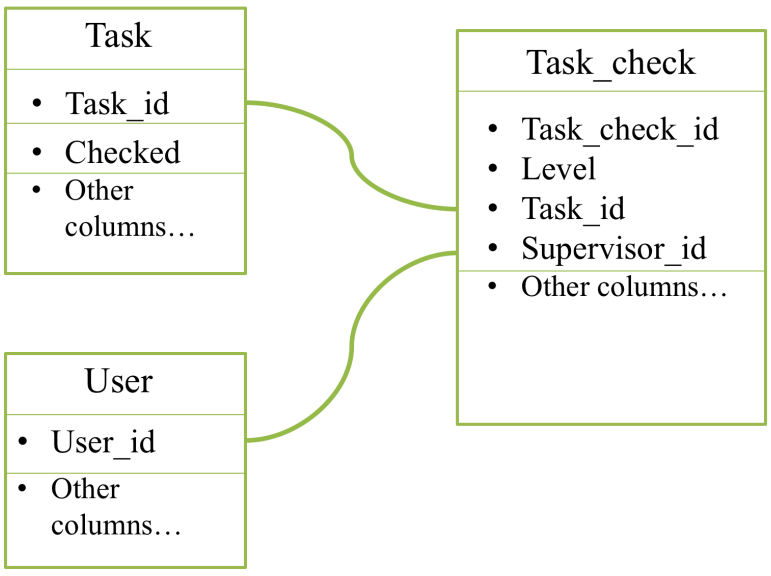
За допомогою створення взаємозв'язків «керівник» - «підлеглий» алгоритм формує ієрархічне дерево



Алгоритм розподілення завдань та їх перевірки. Приклад роботи

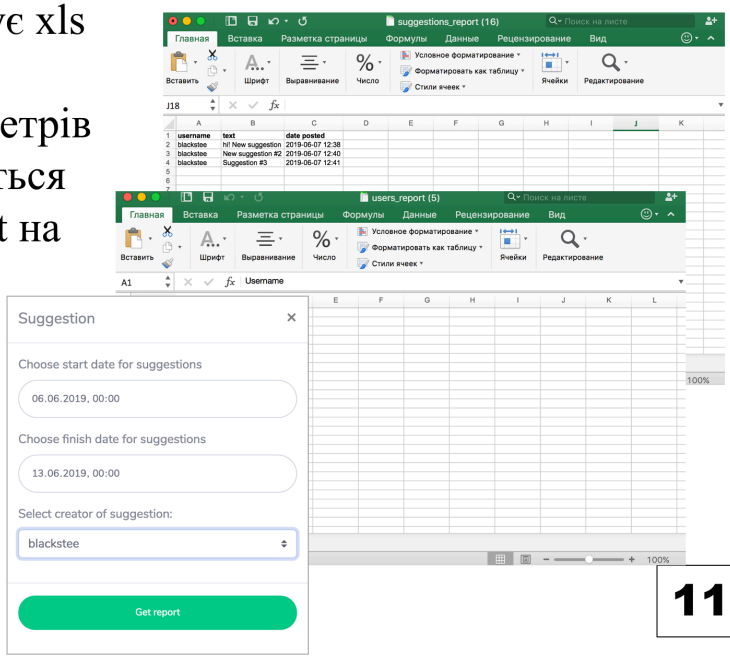


Алгоритм розподілення завдань та їх перевірки. Реалізація у системі



Модуль генерації звітів

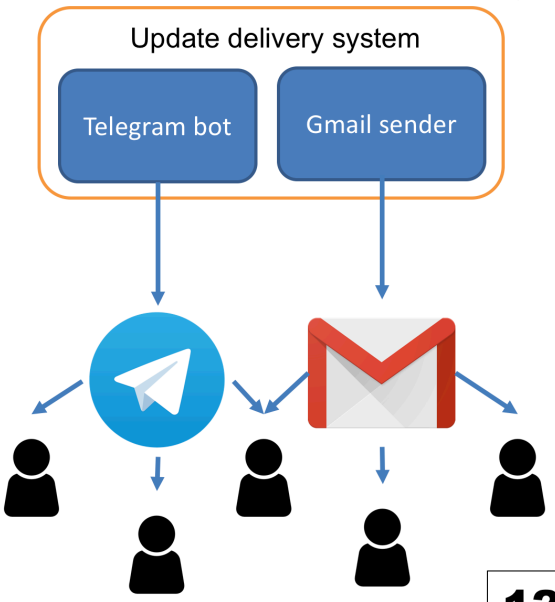
- Модуль генерує xls файл згідно заданих параметрів
- Використовується бібліотека xlwt на Python



Система розсилки повідомлень

Система складається із двох програмних модулів:

- Telegram bot, реалізованого через бібліотеку telegram на Python
- Модуль відправки повідомлень через email - із використанням бібліотеки django.core.email

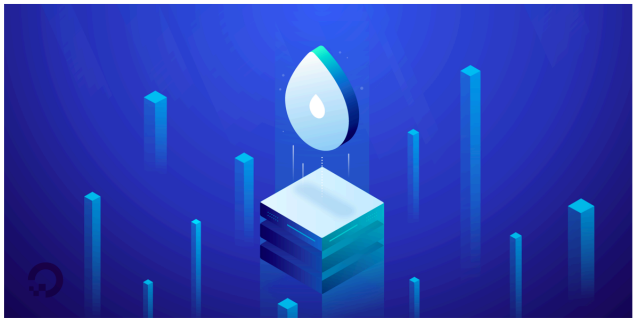


Модуль авторизації

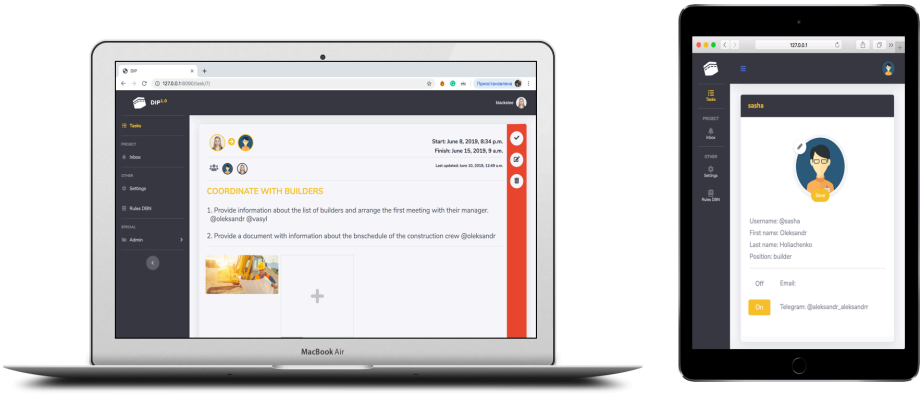
Авторизація виконується за допомогою вбудованої Django authentication system, що дозволяє підтримувати відповідний рівень конфіденційності, включаючи відслідковування сесій.

Хостинг

Хостинг серверу та бази даних відбувається через хмарний сервер на Digital Ocean.
Операційна система: Ubuntu.



Аналіз розробленої системи



Тестування

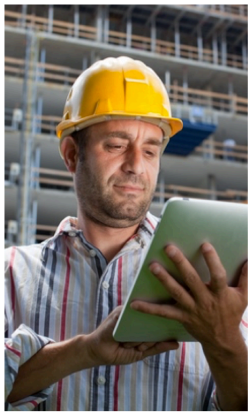
- Було проведено мануальне тестування коду та внесено відповідні вдосконалення
- Здійснено тестування стабільності роботи та захисту даних
- Було проведено базове тестування реальними користувачами та виправлено недоліки
- Система впроваджуватиметься у будівельно-інженерній компанії та тестування продовжуватиметься

Порівняння із існуючими аналогами

	Гранд-смета	АВК-5	DIP
Робота із документацією	✓	✓	✓
Різні рівні доступу			✓
Кросплатформність			✓
Доступ до ДБН	✓		✓
Комунікаційний модуль			✓
Безпека системи		✓	✓

Рекомендації для подальшого вдосконалення

- робота із проектно-кошторисною документацією, включаючи креслення та можливість внесення змін;
- мессенджер для працівників;
- можливість прикріплення елементів проектно-кошторисної документації та ДБН до завдань та коментарів у системі;
- автоматизований кошторис.



ВИСНОВКИ

- Було проведено аналіз існуючих проблем у будівельно-інженерній галузі, розглянуто представлені на ринку аналоги та відповідно сформовано вимоги до розроблюваної системи у вигляді функціональних, нефункціональних вимог та вимог до безпеки.
- Після порівняння засобів реалізації було обрано створення програмного забезпечення у вигляді WEB-застосунку з мобільною WEB-версією мовою програмування Python за допомогою фреймворку Django із використанням СКБД PostgreSQL.
- Було створено програмне забезпечення “DIP” для інженерно-будівельних компаній у вигляді WEB-застосунку з мобільною WEB-версією.
- Розробка програмного забезпечення виконана у повному обсязі та відповідає поставленим вимогам. Тестування системи виконано у відповідності до затвердженої програми та методики тестування.
- Програмне забезпечення “DIP” є універсальним та доступним інструментом для систематизації та контролю усіх процесів у компанії.

НАГОРОДИ ТА ВІДЗНАКИ

Sikorsky Challenge 2018


НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

НАУКОВИЙ ПАРК «КИЇВСЬКА ПОЛІТЕХНІКА»

ІННОВАЦІЙНА ЕКОСИСТЕМА «SIKORSKY CHALLENGE»

ДИПЛОМ ФІНАЛІСТА

Цей диплом засвідчує, що команда проекту

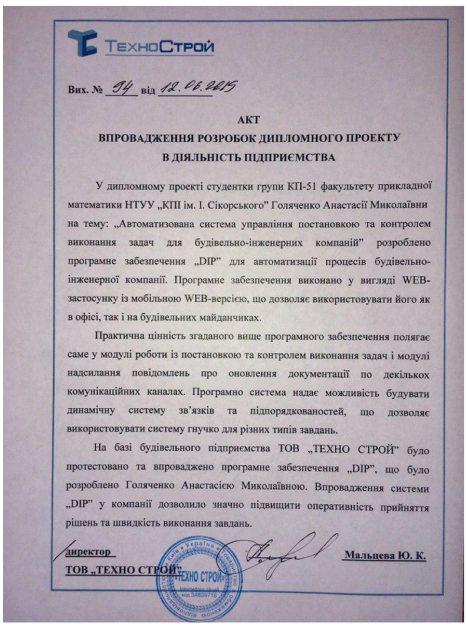
DIP -Data In Pocket, програмне забезпечення, створене спеціально для співробітників сучасних інженерно-будівельних компаній

стала фіналістом Конкурсу стартапів
VII Фестивалю інноваційних проектів

“SIKORSKY CHALLENGE 2018”

Президент Наукового парку
«Київська політехніка» М. Згуровський

АКТ ВПРОВАДЖЕННЯ



ТехноСтрой

Вих. № 94 від 12.02.2015

АКТ
ВПРОВАДЖЕННЯ РОЗРОБОК ДИПЛОМНОГО ПРОЕКТУ
В ДІЯЛЬНІСТЬ ПІДПРИЄМСТВА

У дипломному проекті студентки групи КП-51 факультету прикладної математики НТУУ „КПІ ім. І. Сікорського" Голяченко Анастасії Миколаївни на тему: „Автоматизована система управління постановкою та контролем виконання задач для будівельно-інженерних компаній" розроблено програмне забезпечення „DIP" для автоматизації процесів будівельно-інженерної компанії. Програмне забезпечення виконано у вигляді WEB-застосунку із мобільною WEB-версією, що дозволяє використовувати його як в офісі, так і на будівельних майданчиках.

Практична цінність згаданого вище програмного забезпечення полягає саме у модулі роботи із постановкою та контролем виконання задач і модулі надсилання повідомлень про оновлення документації по декількох комунікаційних каналах. Програмно система надає можливість будувати динамічну систему зв'язків та підпорядкованостей, що дозволяє використовувати систему гнучко для різних типів завдань.

На базі будівельного підприємства ТОВ „ТЕХНО СТРОЙ" було протестовано та впроваджено програмне забезпечення „DIP", що було розроблено Голяченко Анастасією Миколаївною. Впровадження системи „DIP" у компанії дозволило значно підвищити оперативність прийняття рішень та швидкість виконання завдань.

директор ТОВ „ТЕХНО СТРОЙ" Мальцева Ю. К.

Вже впроваджено у
ТОВ “ТЕХНО СТРОЙ”

ТехноСтрой

СТАТТЯ

Lyushenko L., Holiachenko A. (2020) Optimization of the Method of Technical Analysis of Cryptocurrency Price Differences Movements. In: Hu Z., Petoukhov S., Dychka I., He M. (eds) Advances in Computer Science for Engineering and Education II. ICCSEEA 2019. Advances in Intelligent Systems and Computing, vol 938. Springer, Cham

Стаття у Springer

2019
Conference Speech

The certificate is awarded to

Anastasiia Holiachenko

Give a conference speech

Optimization of the Method of Technical Analysis of Cryptocurrency Price Differences Movements

The Second International Conference on Computer Science, Engineering and Education Applications (ICCSEEA2019)

26-27 January 2019, Kiev, Ukraine

Conference Chair

January 26, 2019
Date

Springer

Виступ на конференції
ICCSEEA2019

Дякую за увагу!

Додаток 3

Лістинг програми

1.1 Перша міграція до бази даних

```
class Migration(migrations.Migration):

    initial = True

    dependencies = [
        ('auth', '0008_alter_user_username_max_length'),
    ]

    operations = [
        migrations.CreateModel(
            name='User',
            fields=[
                ('id', models.AutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
                ('password', models.CharField(max_length=128,
verbose_name='password')),
                ('last_login', models.DateTimeField(blank=True,
null=True, verbose_name='last login')),
                ('is_superuser', models.BooleanField(default=False,
help_text='Designates that this user has all permissions without
explicitly assigning them.', verbose_name='superuser status')),
                ('username',
models.CharField(error_messages={'unique': 'A user with that username
already exists.'}, help_text='Required. 150 characters or fewer.
Letters, digits and @/./+/-/_ only.', max_length=150, unique=True,
validators=[django.contrib.auth.validators.UnicodeUsernameValidator()
], verbose_name='username')),
                ('first_name', models.CharField(blank=True,
max_length=30, verbose_name='first name')),
                ('last_name', models.CharField(blank=True,
max_length=30, verbose_name='last name')),
                ('email', models.EmailField(blank=True,
max_length=254, verbose_name='email address')),
                ('is_staff', models.BooleanField(default=False,
help_text='Designates whether the user can log into this admin
site.', verbose_name='staff status')),
                ('is_active', models.BooleanField(default=True,
help_text='Designates whether this user should be treated as active.
Unselect this instead of deleting accounts.',
verbose_name='active')),
                ('date_joined',
models.DateTimeField(default=django.utils.timezone.now,
verbose_name='date joined')),
                ('photo', models.ImageField(blank=True, null=True,
upload_to='')),
                ('user_position', models.TextField(max_length=30,
null=True)),
                ('user_type', models.IntegerField(null=True)),
                ('user_telegram', models.TextField(blank=True,
max_length=30, null=True)),
                ('send_email', models.BooleanField(default=False)),
                ('send_telegram',
models.BooleanField(default=False)),
                ('groups', models.ManyToManyField(blank=True,
help_text='The groups this user belongs to. A user will get all
permissions granted to each of their groups.',
```

```
related_name='user_set', related_query_name='user', to='auth.Group',
verbose_name='groups')),
                ('user_permissions',
models.ManyToManyField(blank=True, help_text='Specific permissions
for this user.', related_name='user_set', related_query_name='user',
to='auth.Permission', verbose_name='user permissions')),
            ],
            options={
                'verbose_name': 'user',
                'verbose_name_plural': 'avatars_us',
                'abstract': False,
            },
            managers=[
                ('objects',
django.contrib.auth.models.UserManager()),
            ],
        ),
        migrations.CreateModel(
            name='Change',
            fields=[
                ('id', models.AutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
                ('date_post', models.DateTimeField(auto_now_add=True,
null=True)),
                ('act_type', models.TextField(null=True)),
                ('before', models.TextField(blank=True, null=True)),
                ('after', models.TextField(blank=True, null=True)),
                ('creator',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='creator_of_change', to=settings.AUTH_USER_MODEL)),
            ],
        ),
        migrations.CreateModel(
            name='Comment',
            fields=[
                ('id', models.AutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
                ('text', models.TextField(max_length=1000,
null=True)),
                ('date_post', models.DateTimeField(auto_now_add=True,
null=True)),
                ('date_update',
models.DateTimeField(auto_now_add=True, null=True)),
                ('updated', models.IntegerField(blank=True,
default=0, null=True)),
                ('author',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='creator_of_comment', to=settings.AUTH_USER_MODEL)),
                ('followers',
models.ManyToManyField(to=settings.AUTH_USER_MODEL)),
            ],
        ),
        migrations.CreateModel(
            name='Link_users',
            fields=[
                ('id', models.AutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
```

```
                ('doer',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='doer', to=settings.AUTH_USER_MODEL)),
                ('supervisor',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='boss', to=settings.AUTH_USER_MODEL)),
            ],
        ),
        migrations.CreateModel(
            name='Pic_comm',
            fields=[
                ('id', models.AutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
                ('picture',
models.ImageField(default='pic_folder/None/no-img.jpg',
upload_to='pic_folder/comments/')),
                ('comment',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
to='dip_manager.Comment')),
            ],
        ),
        migrations.CreateModel(
            name='Pic_Task',
            fields=[
                ('id', models.AutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
                ('picture',
models.ImageField(default='pic_folder/None/no-img.jpg',
upload_to='pic_folder/tasks/')),
            ],
        ),
        migrations.CreateModel(
            name='Pic_User',
            fields=[
                ('id', models.AutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
                ('picture',
models.ImageField(default='pic_folder/None/no-img.jpg',
upload_to='pic_folder/avatars_us/')),
                ('creator',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
to=settings.AUTH_USER_MODEL)),
            ],
        ),
        migrations.CreateModel(
            name='Project',
            fields=[
                ('id', models.AutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
                ('project_name', models.TextField(max_length=100,
null=True)),
                ('description', models.TextField(blank=True,
max_length=10000, null=True)),
            ],
        ),
        migrations.CreateModel(
            name='Suggestion',
            fields=[
```

```
                ('id', models.AutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
                ('date_post', models.DateTimeField(auto_now_add=True,
null=True)),
                ('text', models.TextField(max_length=10000,
null=True)),
                ('creator',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='creator_of_suggestion', to=settings.AUTH_USER_MODEL)),
                ('followers',
models.ManyToManyField(to=settings.AUTH_USER_MODEL)),
            ],
        ),
        migrations.CreateModel(
            name='Task',
            fields=[
                ('id', models.AutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
                ('task_name', models.TextField(max_length=100,
null=True)),
                ('description', models.TextField(blank=True,
max_length=10000, null=True)),
                ('date_start',
models.DateTimeField(auto_now_add=True, null=True)),
                ('date_end', models.DateTimeField(null=True)),
                ('date_update',
models.DateTimeField(auto_now_add=True, null=True)),
                ('updated', models.IntegerField(blank=True,
default=0, null=True)),
                ('checked', models.BooleanField(default=False)),
                ('creator',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='creator_of_task', to=settings.AUTH_USER_MODEL)),
                ('doer',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='doer_of_task', to=settings.AUTH_USER_MODEL)),
                ('followers',
models.ManyToManyField(to=settings.AUTH_USER_MODEL)),
                ('project',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
to='dip_manager.Project')),
            ],
        ),
        migrations.CreateModel(
            name='Task_check',
            fields=[
                ('id', models.AutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
                ('level', models.IntegerField(null=0)),
                ('checked', models.BooleanField(default=False)),
                ('supervisor',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='checks_task', to=settings.AUTH_USER_MODEL)),
                ('task',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='task_being_checked', to='dip_manager.Task')),
            ],
        ),
```

```
        migrations.AddField(
            model_name='pic_task',
            name='task',

field=models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
to='dip_manager.Task'),
        ),
        migrations.AddField(
            model_name='pic_comm',
            name='task',

field=models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='task_of_comm_of_pic', to='dip_manager.Task'),
        ),
        migrations.AddField(
            model_name='comment',
            name='task',

field=models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='related_task', to='dip_manager.Task'),
        ),
    ]
```

1.2 Моделі

```
class ProjectManager(models.Manager):


    def create_project(self, project_name, description):

        project = self.create(project_name=project_name, description
= description)

        return project




class User(AbstractUser):

    photo = models.ImageField(upload_to = 'pic_folder/comments/',
default = 'pic_folder/comments/def_avatar.jpg', null=False,
blank=False)

    user_position = models.TextField(max_length=30, blank=False,
null=True)

    user_type = models.IntegerField(blank=False, null=True)

    user_telegram = models.TextField(max_length=30, blank=True,
null=True)

    send_email = models.BooleanField(blank=False, null=False, default
= False)

    send_telegram = models.BooleanField(blank=False, null=False,
default = False)
```

```
class Project(models.Model):

    project_name = models.TextField(blank=False, max_length=100,
null=True)

    description = models.TextField(max_length=10000, blank=True,
null=True)

    objects = ProjectManager()




class Task(models.Model):

    task_name = models.TextField(blank=False, max_length=100,
null=True)

    description = models.TextField(max_length=10000, blank=True,
null=True)

    date_start = models.DateTimeField(auto_now_add=True, null=True)

    date_end = models.DateTimeField(blank=False, null=True)

    date_update = models.DateTimeField(auto_now_add=True, null=True)

    updated = models.IntegerField(blank=True, default=0, null=True)

    creator = models.ForeignKey(User, on_delete=models.CASCADE,
related_name="creator_of_task")

    doer = models.ForeignKey(User, on_delete=models.CASCADE,
related_name="doer_of_task")

    followers = models.ManyToManyField(User)

    project = models.ForeignKey(Project, on_delete=models.CASCADE)

    checked = models.BooleanField(blank=False, null=False,
default=False)



class CommentManager(models.Manager):


    def create_comment(self, text, creator, task):

        comment = self.create(text=text, author=creator, task=task)

        return comment


class Comment(models.Model):
```

```
    text = models.TextField(max_length=1000, blank=False, null=True)

    author = models.ForeignKey(User, on_delete=models.CASCADE,
related_name="creator_of_comment")

    date_post = models.DateTimeField(auto_now_add=True, null=True)

    date_update = models.DateTimeField(auto_now_add=True, null=True)

    updated = models.IntegerField(blank=True, default=0, null=True)

    followers = models.ManyToManyField(User)

    task = models.ForeignKey(Task, on_delete=models.CASCADE,
related_name="related_task")

    objects = CommentManager()



class Suggestion(models.Model):


    date_post = models.DateTimeField(auto_now_add=True, null=True)

    creator = models.ForeignKey(User, on_delete=models.CASCADE,
related_name="creator_of_suggestion")

    text = models.TextField(max_length= 10000, blank=False,
null=True)

    followers = models.ManyToManyField(User)



class Change (models.Model):

    date_post = models.DateTimeField(auto_now_add=True, null=True)

    creator = models.ForeignKey(User, on_delete=models.CASCADE,
related_name="creator_of_change")

    act_type = models.TextField(blank=False, null=True)

    before = models.TextField(blank=True, null=True)

    after = models.TextField(blank=True, null=True)



class Pic_comm (models.Model):

    task = models.ForeignKey(Task, on_delete=models.CASCADE,
related_name="task_of_comm_of_pic")

    comment = models.ForeignKey(Comment, on_delete=models.CASCADE)
```

```
    picture = models.ImageField(upload_to = 'pic_folder/comments/',
default = 'pic_folder/None/no-img.jpg')



class Pic_Task(models.Model):

    task = models.ForeignKey(Task, on_delete=models.CASCADE)

    picture = models.ImageField(upload_to='pic_folder/tasks/',
default='pic_folder/None/no-img.jpg')



class Pic_User(models.Model):

    creator = models.ForeignKey(User, on_delete=models.CASCADE)

    picture = models.ImageField(upload_to='pic_folder/avatars_us/',
default='pic_folder/None/no-img.jpg')


class Link_users(models.Model):

    supervisor = models.ForeignKey(User, on_delete=models.CASCADE,
related_name="boss")

    doer = models.ForeignKey(User, on_delete=models.CASCADE,
related_name="doer")


class Task_check (models.Model):

    supervisor = models.ForeignKey(User, on_delete=models.CASCADE,
related_name="checks_task")

    task = models.ForeignKey(Task, on_delete=models.CASCADE,
related_name="task_being_checked")

    level = models.IntegerField(blank=False, null = 0)

    checked = models.BooleanField(blank= False, null=False, default =
False)
```

2.1 Telegram bot

```
class Bot:
    def __init__ (self):

        self.my_token = my_token



    def send(self, msg, chat_id):

    self.bot = telegram.Bot(token=self.my_token)
    self.bot.sendMessage(chat_id=chat_id, text=msg)
```

```
    def get_chat_id (self, username):

        URL =
"https://api.telegram.org/bot"+self.my_token+"/getUpdates"

        r = requests.get(url=URL)

        data = r.json()

        for i in data['result']:
            if 'username' in i['message']['from']:
                if i['message']['from']['username'] == username:
                    return i['message']['chat']['id']
```

3.1 Модуль генерації звітів

```
# content-type of response
    response = HttpResponse(content_type='application/ms-excel')

    # decide file name
    response['Content-Disposition'] = 'attachment;
filename="users_report.xls"'

    # creating workbook
    wb = xlwt.Workbook(encoding='utf-8')

    # adding sheet
    ws = wb.add_sheet("users")

    # Sheet header, first row
    row_num = 0

    font_style = xlwt.XFStyle()
    # headers are bold
    font_style.font.bold = True

    # column header names
    columns = ['Username', 'First name', 'Last name', 'Position']

    # write column headers in sheet
    for col_num in range(len(columns)):
        ws.write(row_num, col_num, columns[col_num], font_style)

    # Sheet body, remaining rows
    font_style = xlwt.XFStyle()

    # get your data

    data = User.objects.all()

    for my_row in data:
        row_num = row_num + 1
        ws.write(row_num, 0, my_row.username, font_style)
        ws.write(row_num, 1, my_row.first_name, font_style)
        ws.write(row_num, 2, my_row.last_name, font_style)
        ws.write(row_num, 3, my_row.user_position, font_style)
```

Додаток 4

Акт впровадження та диплом

ТехноСтрой

Вих. № 94 від 12.06.2015

АКТ

ВПРОВАДЖЕННЯ РОЗРОБОК ДИПЛОМНОГО ПРОЕКТУ

В ДІЯЛЬНІСТЬ ПІДПРИЄМСТВА

У дипломному проекті студентки групи КП-51 факультету прикладної математики НТУУ „КПІ ім. І. Сікорського" Голяченко Анастасії Миколаївни на тему: „Автоматизована система управління постановкою та контролем виконання задач для будівельно-інженерних компаній" розроблено програмне забезпечення „DIP" для автоматизації процесів будівельно-інженерної компанії. Програмне забезпечення виконано у вигляді WEB-застосунку із мобільною WEB-версією, що дозволяє використовувати його як в офісі, так і на будівельних майданчиках.

Практична цінність згаданого вище програмного забезпечення полягає саме у модулі роботи із постановкою та контролем виконання задач і модулі надсилання повідомлень про оновлення документації по декількох комунікаційних каналах. Програмно система надає можливість будувати динамічну систему зв'язків та підпорядкованостей, що дозволяє використовувати систему гнучко для різних типів завдань.

На базі будівельного підприємства ТОВ „ТЕХНО СТРОЙ" було протестовано та впроваджено програмне забезпечення „DIP", що було розроблено Голяченко Анастасією Миколаївною. Впровадження системи „DIP" у компанії дозволило значно підвищити оперативність прийняття рішень та швидкість виконання завдань.

директор
ТОВ „ТЕХНО СТРОЙ" Мальцева Ю. К.

Україна Київ Товариство обмеженою відповідальністю ТЕХНО СТРОЙ код 34839716

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

НАУКОВИЙ ПАРК «КИЇВСЬКА ПОЛІТЕХНІКА»

ІННОВАЦІЙНА ЕКОСИСТЕМА «SIKORSKY CHALLENGE»

Sikorsky Challenge

ДИПЛОМ ФІНАЛІСТА

Цей диплом засвідчує, що команда проекту

DIP -Data In Pocket, програмне забезпечення, створене спеціально для співробітників сучасних інженерно-будівельних компаній

стала фіналістом Конкурсу стартапів
VII Фестивалю інноваційних проектів

"SIKORSKY CHALLENGE 2018"

Президент Наукового парку
«Київська політехніка» М. Згуровський

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

АВТОМАТИЗОВАНА СИСТЕМА УПРАВЛІННЯ ПОСТАНОВКОЮ ТА КОНТРОЛЕМ ВИКОНАННЯ ЗАДАЧ ДЛЯ БУДІВЕЛЬНО-ІНЖЕНЕРНИХ КОМПАНІЙ

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проекту:

_____ Л.А. Люшенко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ А.М. Голяченко

2018

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Автоматизована система управління постановкою та контролем виконання задач для будівельно-інженерних компаній систематизує усі процеси, що стосуються надання та контролю виконання завдань у будівельно-інженерних компаніях, та надає можливість створення індивідуального ієрархічного дерева. Система виконана у вигляді WEB-застосунку із мобільною WEB-версією.

2. МЕТА ТЕСТУВАННЯ

Під час тестування мають бути перевірені наступні елементи:

- коректність роботи модуля надання задач;
- коректність роботи модуля контролю виконання задач;
- коректність роботи модуля створення ієрархічного дерева “керівник – підлеглий”;
- коректність роботи журналів;
- відповідність етапів перевірки задач ієрархічним взаємозв’язкам;
- зручність інтерфейсу.

3. МЕТОДИ ТЕСТУВАННЯ

Виконуються модульне, інтеграційне та системне тестування.

- Мета модульного тестування: перевірка коректності роботи окремих модулів, що забезпечують відповідно:
  - надання задач та розсилання повідомлень;
  - контроль виконання задач;
  - створення ієрархічного дерева “керівник – підлеглий”;
  - робота журналів пропозицій та змін у системі;
  - інтерфейс користувача.

- Мета інтеграційного тестування: перевірка відповідності етапів підтвердження задач ієрархічним взаємозв’язкам, взаємодія відповідних модулів;
- Мета системного тестування: перевірка своєчасності та правильності взаємодії СКБД та розробленої системи та відповідність інтерфейсу користувача.

Використовуються наступні методи:

- ручне тестування інтерфейсу;
- функціональне тестування;
- тестування взаємодії.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально. Перевірка роботи розробленої системи здійснюється наступним чином

1. Тестування коректної роботи модулю надання задач та розсилання повідомлень шляхом перевірки правильності результатів роботи із набором тестових даних.
2. Тестування коректної роботи модулю контролю виконання задач шляхом перевірки правильної роботи набору тестових даних.
3. Тестування коректності роботи модулю створення ієрархічних зв’язків та відповідності під час взаємодії із іншими модулями.
4. Статичне тестування коду.
5. Тестування інтерфейсу користувача.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

АВТОМАТИЗОВАНА СИСТЕМА УПРАВЛІННЯ ПОСТАНОВКОЮ ТА КОНТРОЛЕМ ВИКОНАННЯ ЗАДАЧ ДЛЯ БУДІВЕЛЬНО-ІНЖЕНЕРНИХ КОМПАНІЙ

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проекту:

_____ Л.А. Люшенко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ А.М. Голяченко

2019

ЗМІСТ

1. ОПИС СТРУКТУРИ ІНТЕРФЕЙСУ WEB-ЗАСТОСУНКУ

Розроблене програмне забезпечення для будівельно-інженерних компаній у вигляді WEB-застосунку із мобільною WEB-версією складається не має статичних сторінок. Усі сторінки завантажуються відповідно до того, хто з користувачів увійшов у систему та до якого типу він відноситься:

- звичайний користувач;
- менеджер;
- топ-менеджер;
- адміністратор.

Були розроблені такі сторінки:

- Index – головна сторінка застосунку. Якщо на неї перейшов користувач, що вже знаходиться у системі, тоді вона переадресовує його на сторінку Tasks відповідно до його акаунту. Якщо на головну сторінку зайшов новий користувач, він може зайти у систему у автоматично відкритому вікні входу у систему.
- Tasks – на цю сторінку може перейти лише авторизований користувач. Зліва розташована панель швидкого доступу до різних розділів відповідно до типу користувача:
  - Tasks (активна сторінка);
  - Inbox;
  - Settings;
  - Suggestions;
  - Admin;
  - Rules.
  - На верхній панелі вказані:
    - ім’я користувача;
    - фото користувача у системі.

При натисненні на фото користувача з’являється випадаючий список, де користувач може вийти із системи (лише після підтвердження у додатковому вікні про бажання вийти із системи), або перейти на сторінку свого акаунту Settings.

- Inbox – сторінка із повідомленнями про створення завдання самим користувачем, чи на які він був назначений або відмічений у форматі “@ім’я_користувача” в тексті самого завдання чи у коментарях.
- Settings – сторінка користувача, де він може змінити дані свого профілю.
- Suggestions – для звичайного користувача це сторінка із однією кнопкою “Подати пропозицію”, натиснувши на яку користувач може ввести текст пропозиції, відмітити (за бажанням) користувачів у форматі “@ім’я_користувача”, після чого відправити свою пропозицію. Для користувачів типу менеджер ця сторінка відображає пропозиції, що надходили від працівників, є можливість надіслати свою пропозицію та автоматично згенерувати звіт по окремим критеріям (дата, користувачі).
- Admin – сторінка, доступна лише для користувача типу адміністратор.
- Rules – сторінка, доступна для усіх користувачів, являє собою пошукову систему по ДБН.

2. РОБОТА ІЗ АКАУНТОМ КОРИСТУВАЧА

2.1. Процедура авторизації користувача

Авторизація користувача відбувається на сторінці входу у систему (див. рис. 1). Користувач має ввести своє ім’я користувача та пароль у системі, які йому надав адміністратор (за умови, що користувач авторизується перший раз та ще не змінював своє ім’я користувача та пароль).

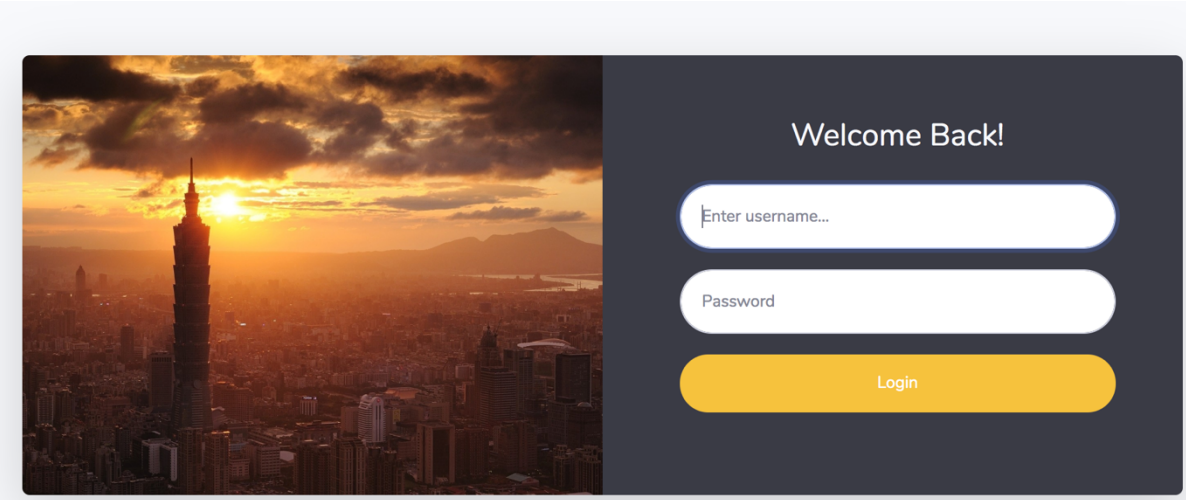


Рис. 1. Сторінка входу у систему

У випадку, коли користувач намається увійти із пустими даними у полях, система повідомить про помилку (див. рис. 2).

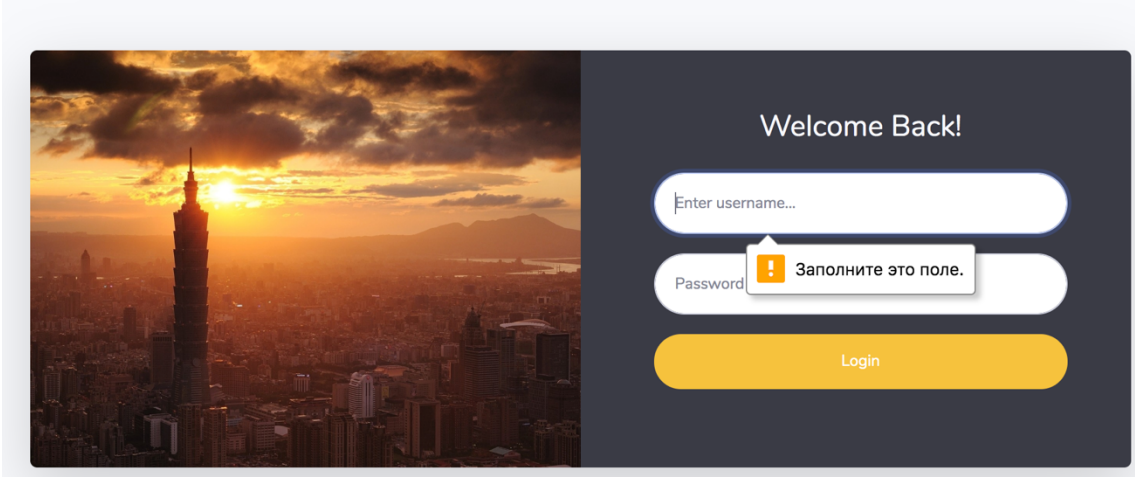


Рис. 2. Повідомлення про помилку на сторінці входу у систему

У разі успішної авторизації система перенаправить користувача до сторінки Tasks.

2.2. Процедура виходу із системи користувача

Для виходу із системи користувач має натиснути на зображення свого профілю на верхній панелі. Після чого відкриється випадаючий список, де є варіант вибору “Log out” (див. рис. 3). Користувач має настиснути на цю кнопку. Відкриється модальне вікно із питанням, чи користувач впевнений, що хоче вийти із системи.

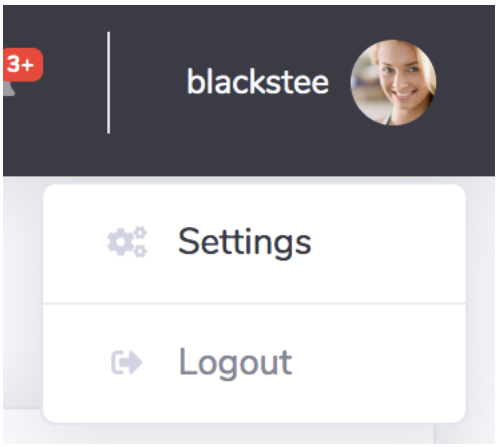


Рис. 3. Випадаюче меню у верхній панелі

У випадку, якщо користувач передумав, він може закрити модальне вікно і далі працювати у системі.

Якщо користувач все одно хоче вийти із системи, він має натиснути червону кнопку “Log out” у модальному вікні (див. рис. 4). Система автоматично закінчить сесію даного користувача та перенаправить його на сторінку входу у систему.

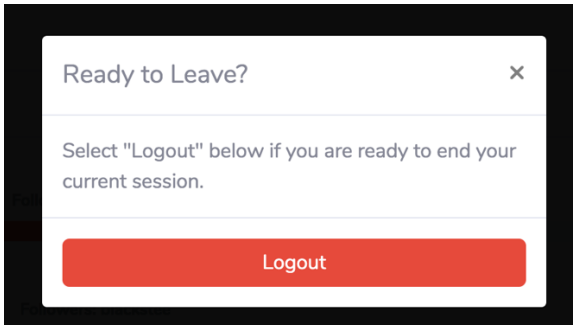


Рис. 4. Модальне вікно виходу із системи

2.3. Налаштування профілю користувача

Кожен користувач може перейти до сторінки налаштувань профілю та змінити своє фото, ім’я користувача у системі, повне ім’я, назву посади, електронну пошту та ім’я у Telegram (див. рис. 5). Також він може налаштувати розсилку повідомлень про оновлення через електронну пошту та Telegram.

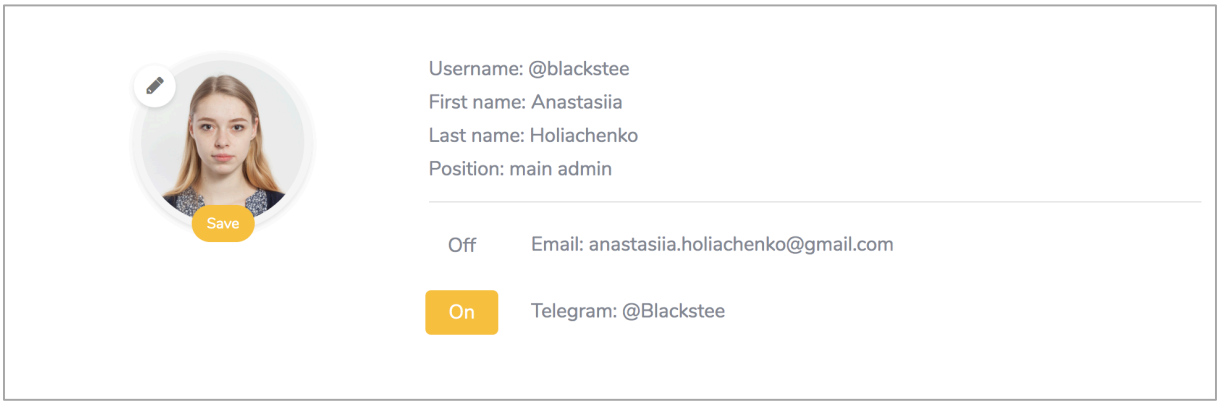


Рис. 5. Сторінка налаштувань профілю

3. РОБОТА ІЗ ЗАВДАННЯМИ

3.1. Робота зі списком завдань

У випадку, коли користувач авторизований у системі, він може перейти до сторінки Tasks. Користувачі типу “Менеджер”, “Топ-менеджер” та “Адміністратор” матимуть додаткову кнопку “New task” (див. рис. 6), для користувача типу “Звичайний користувач” на сторінці буде лише список із завданнями.

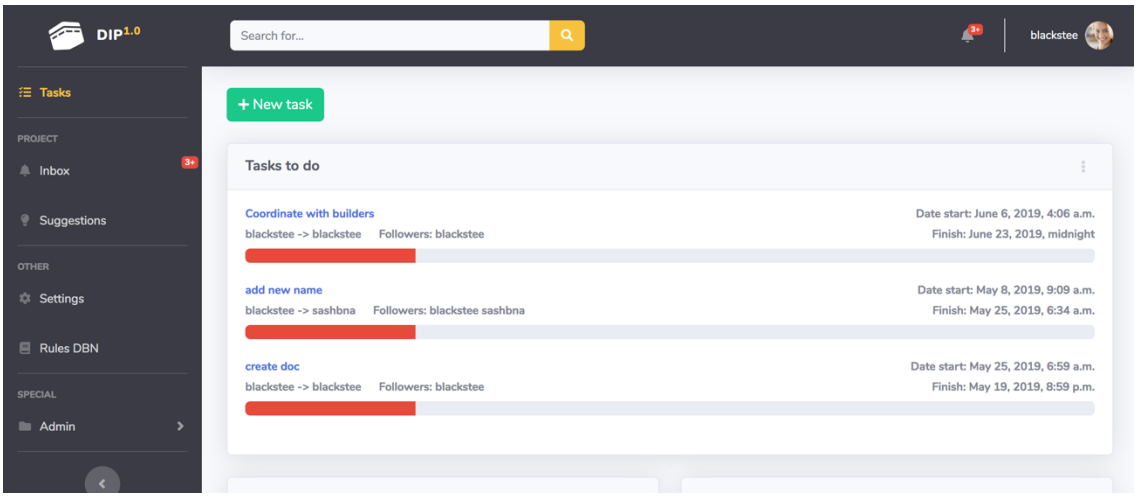


Рис. 6. Сторінка із завданнями

Користувач будь-якого типу може перейти на сторінку завдання, якщо він зв’язаний з ним (є автором, є виконавцем чи був відмічений у завданні).

Користувачі типу “Менеджер”, “Топ-менеджер” та “Адміністратор” можуть створити нове завдання натиснувши на кнопку “New task” (див. рис. 7).

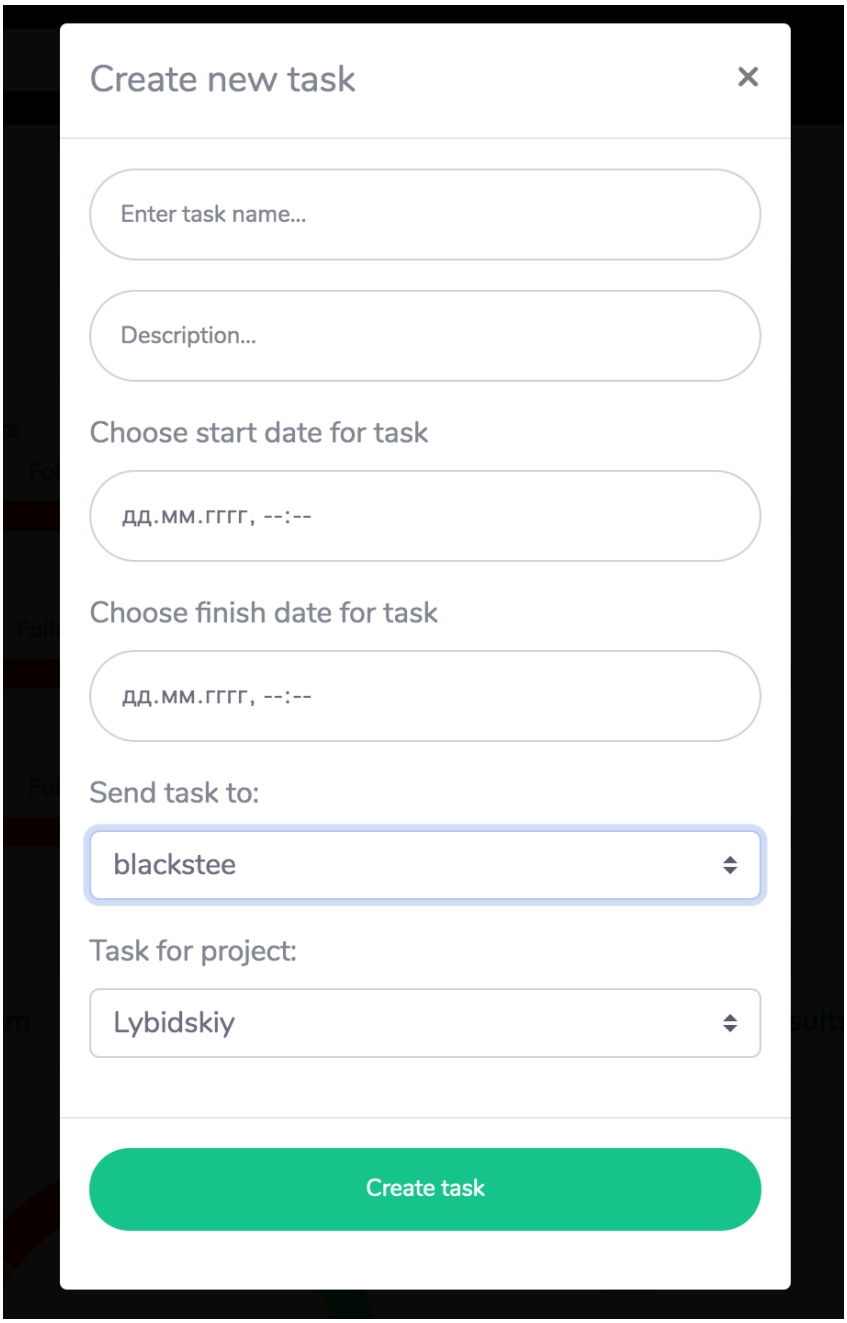


Рис. 7. Модальне вікно створення нового завдання

3.2. Робота із завданням

На сторінці завдання розміщена картка із інформацією про завдання (див. рис. 8).

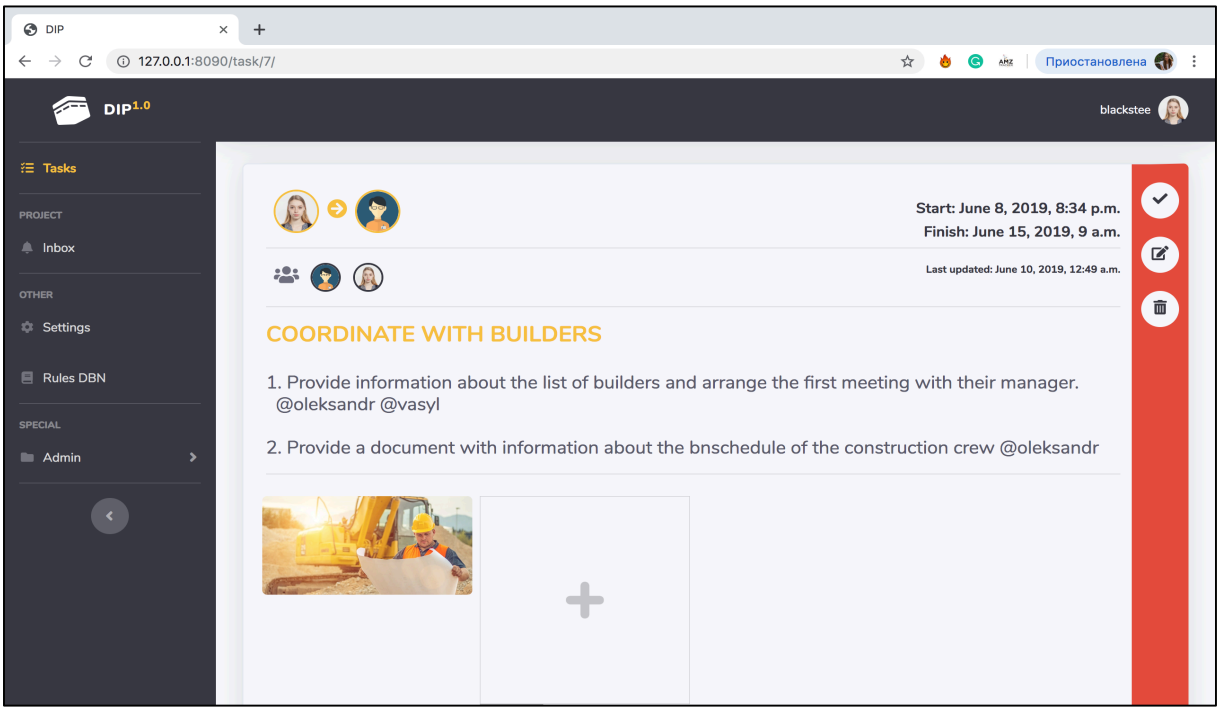


Рис. 8. Сторінка завдання

Колір картки справа (бокова панель) показує у якому статусі знаходиться завдання. Також на ній розміщені дві кнопки – “Увімкнути/вимкнути режим редагування завдання” та “Видалити завдання”. Дані кнопки керування завданням доступні лише користувачу, який є автором цього завдання (див. рис. 9).

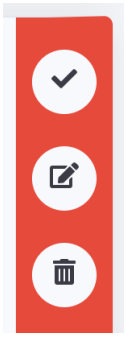


Рис. 9. Панель інструментів завдання

При натисненні на кнопку видалення користувач спочатку має підтвердити свій намір видалити саме це завдання у відкритому модальному вікні. При виборі кнопки редагування завдання вона змінює свій колір, знизу вікна з’являється повідомлення про те, що режим редагування завдання активований. Після цього користувач може вносити

зміни до завдання та наприкінці має знову натиснути кнопку редагування, щоб зберегти усі зміни та вийти із режиму редагування. Якщо завдання було відредаговано один раз, під датою кінцевого терміну виконання завдання з'являється дата останнього оновлення завдання (див. рис. 10).

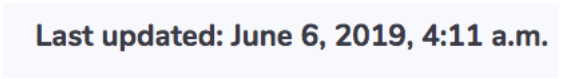


Рис. 10. Дата останнього оновлення завдання

Користувачі будь-якого типу можуть залишати коментарі під завданнями, із якими вони зв'язані. Для цього вони мають прокрутити сторінку завдання вниз до поля вводу коментаря (див. рис. 11). Також користувач може додати фото до коментаря.

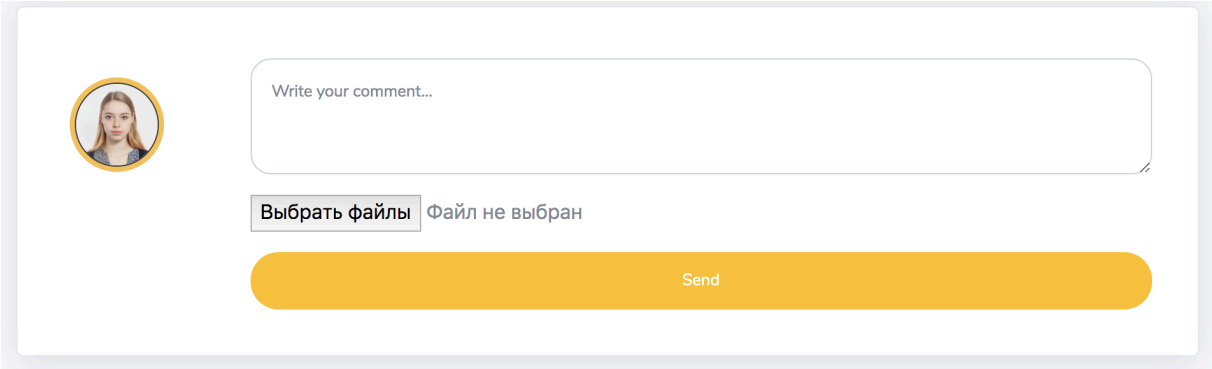


Рис. 11. Поле вводу коментаря під завданням

Ще нижче ніж поле вводу коментаря розташовані коментарі, що залишили усіх користувачі стосовно цього завдання. Якщо коментар належить даному авторизованому користувачу, він має можливість видалити його натиснувши на кнопку видалення у вигляді смітника (див. рис. 12).

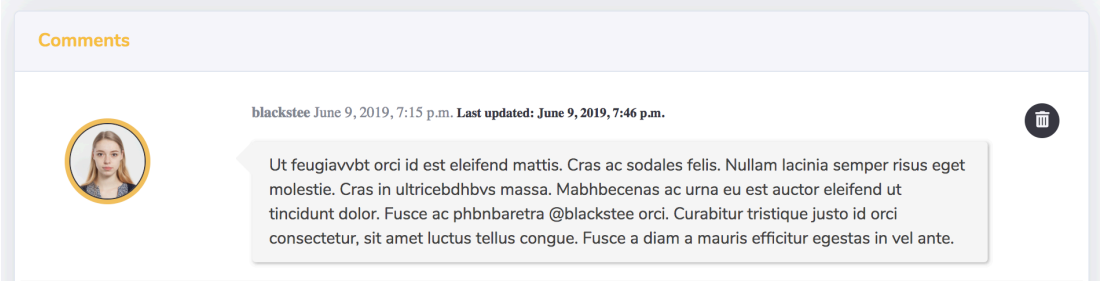


Рис. 12. Відображення коментарів під завданням

Також кожен користувач може внести зміни до свого коментаря. Для цього йому достатньо клікнути на текст коментаря і режим редагування коментаря буде увімкнено. Після внесення змін до коментаря, поряд з’явиться жовта кнока у вигляді повітряного літака “Відправити” (див. рис. 13). Після натискання даної кнопки зміни внесені до коментаря будуть збережені.

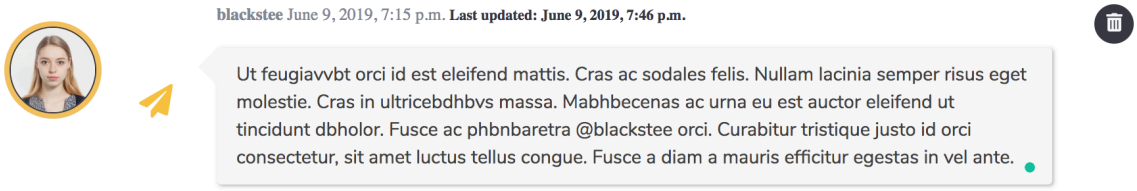


Рис. 13. Коментар користувача у режимі внесення змін

Після першого редагування коментаря на ньому з’являється помітка про останнє редагування із датою та часом внесення змін (див. рис. 14).

Рис. 14. Заголовок коментаря із поміткою про останнє редагування

4. ПАНЕЛЬ АДМІНІСТРАТОРА

4.1. Меню адміністратора

Панель адміністратора доступна лише користувачам типу “Адміністратор”. Користувач може обрати один із запропонованих варіантів для подальшої роботи (див. рис. 15).

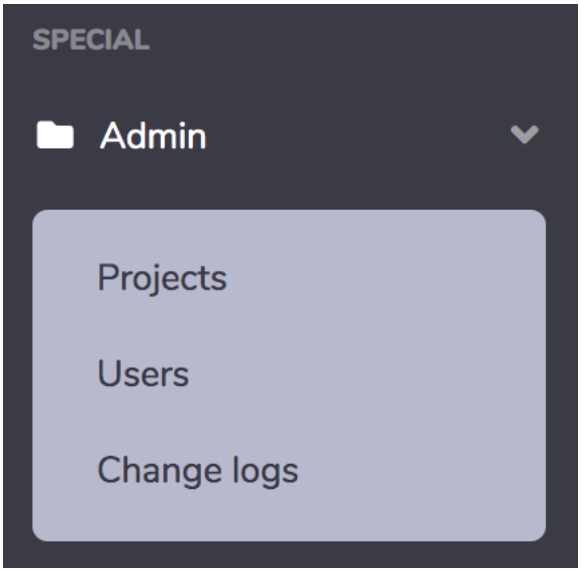


Рис. 15. Меню панелі адміністратора

4.2. Робота із проектами

Натиснувши на кнопку “Projects” користувач перенаправляється на сторінку проектів (див. рис. 16).

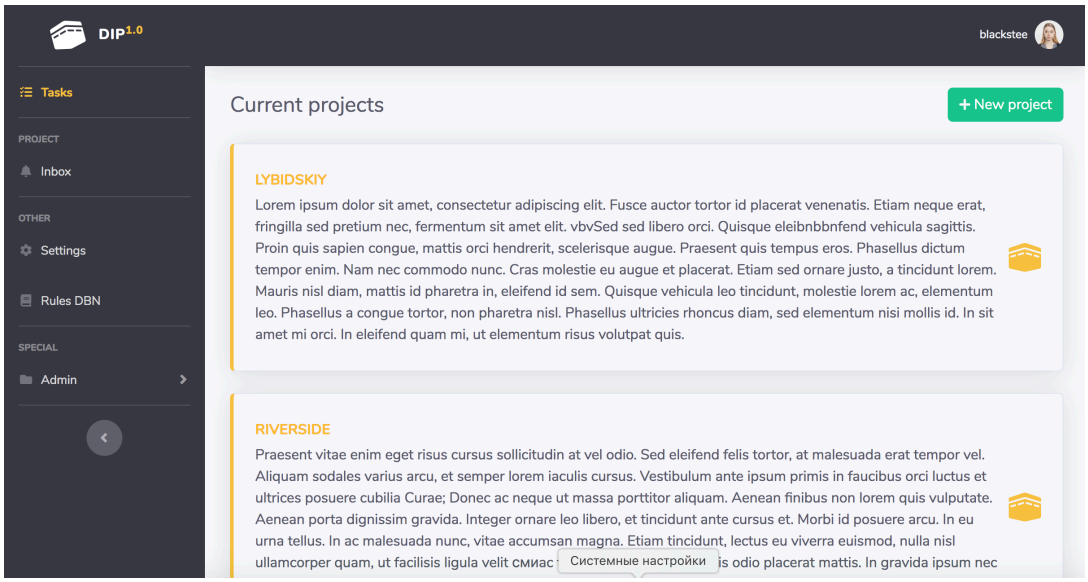


Рис. 16. Список проектів на панелі адміністратора

Користувач може створити новий проект натиснувши на відповідну кнопку “New project” (див. рис. 17). Для цього він має додати інформацію про новий проект у модальному вікні створення проекту.

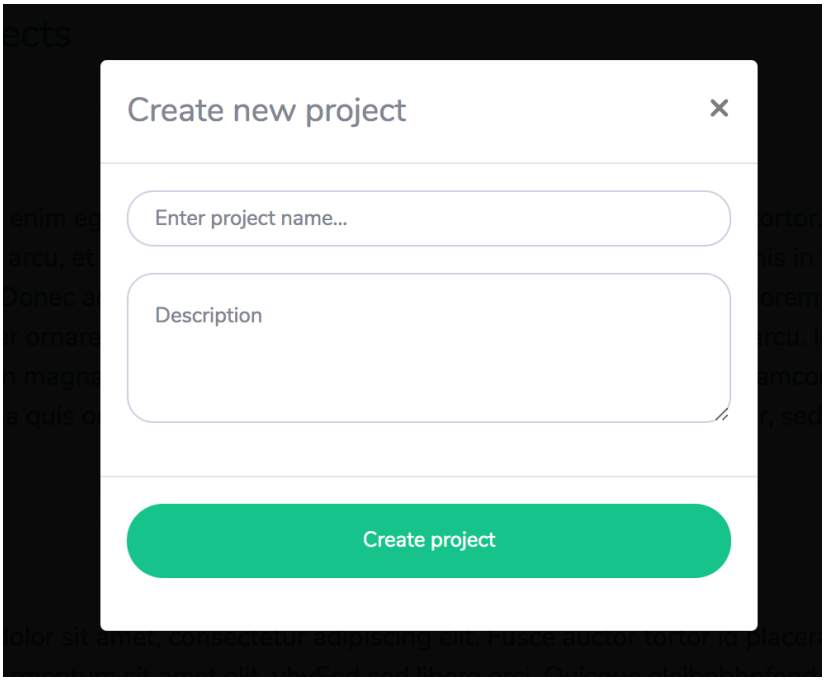


Рис. 17. Модальне вікно створення проекту

Також користувач може перейти на сторінку кожного проекту окремо (див. рис. 18). На панелі інструментів доступна кнопка видалення проекту. Для цього потрібно її натиснути та підтвердити у відкритому модальному вікні своє бажання видалити саме цей проект.

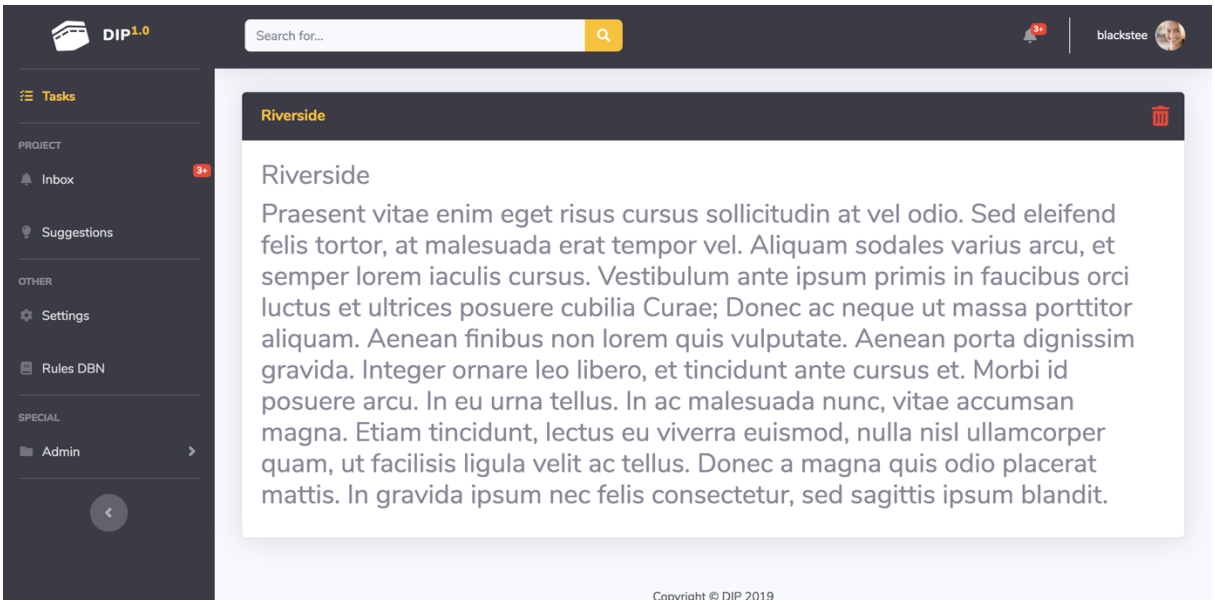


Рис. 18. Сторінка проекту

Для редагування інорфмації про проект користувач має клікнути на інформацію, яку він хоче змінити, автоматично увімкнеться режим редагування проекту. Після внесення першої зміни на панелі інструментів з’явиться кнопка підтвердження змін у вигляді жовтого паперового літака (див. рис. 19). Натиснувши на неї, користувач зберігає внесені до проекту зміни.

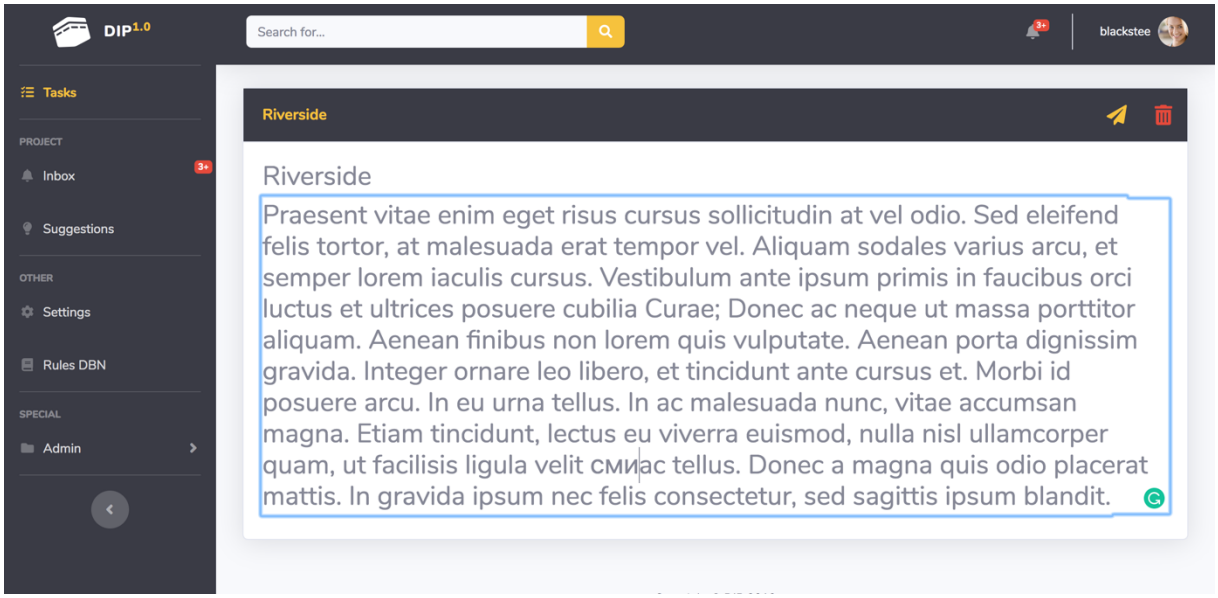


Рис. 19. Режим редагування проекту

4.3. Робота із користувачами

Натиснувши на кнопку “Users” користувач перенаправляється на сторінку користувачів (див. рис. 20).

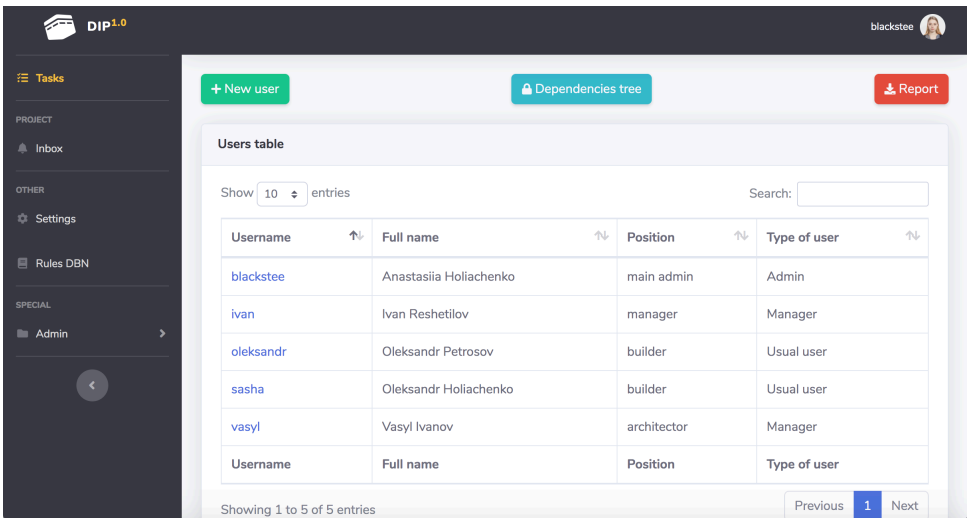


Рис. 20. Список користувачів на панелі адміністратора

Адміністратор може створити нового користувача натиснувши на відповідну кнопку “New user” (див. рис. 21). Для цього він має додати інформацію про нового користувача у модальному вікні створення користувача.

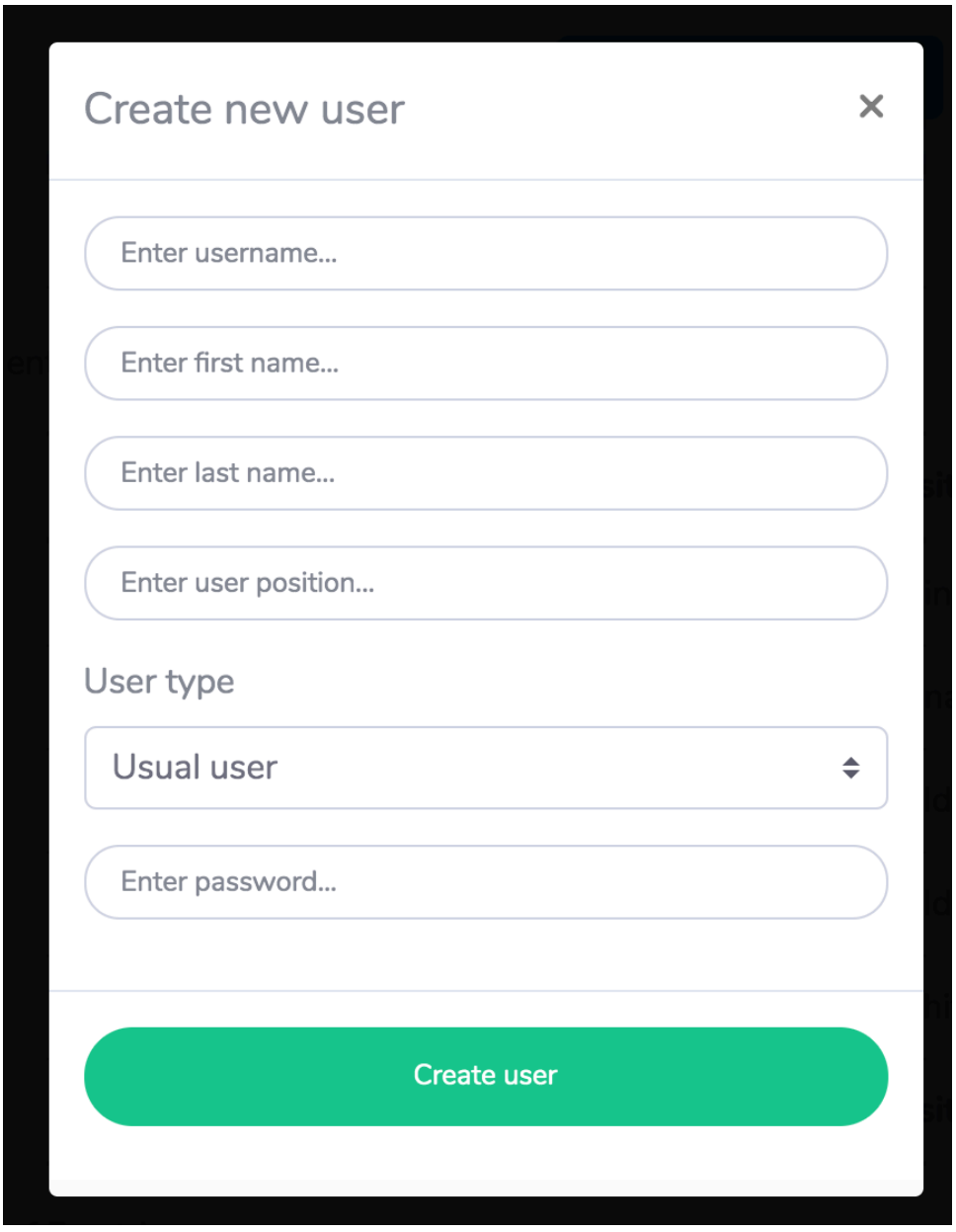


Рис. 21. Модальне вікно створення користувача

Також він може перейти на сторінку кожного користувача окремо (див. рис. 22). На панелі інструментів доступна кнопка видалення користувача. Для цього потрібно її натиснути та підтвердити у відкритому модальному вікні своє бажання видалити саме цього користувача.

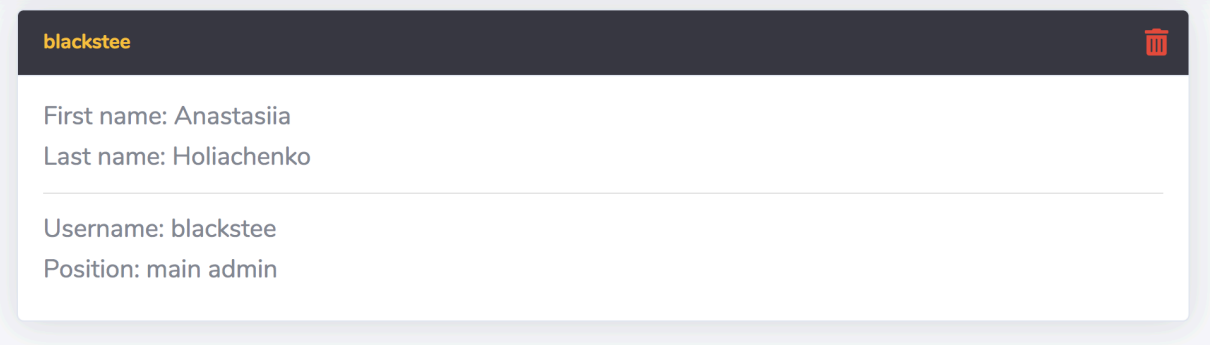


Рис. 22. Сторінка користувача

Для редагування інорфмації про користувача Адміністратор має клікнути на інформацію, яку він хоче змінити, автоматично увімкнеться режим редагування користувача. Після внесення першої зміни на панелі інструментів з’явиться кнопка підтвердження змін у вигляді жовтого паперового літака (див. рис. 23). Натиснувши на неї, адміністратор зберігає внесені до користувача зміни.

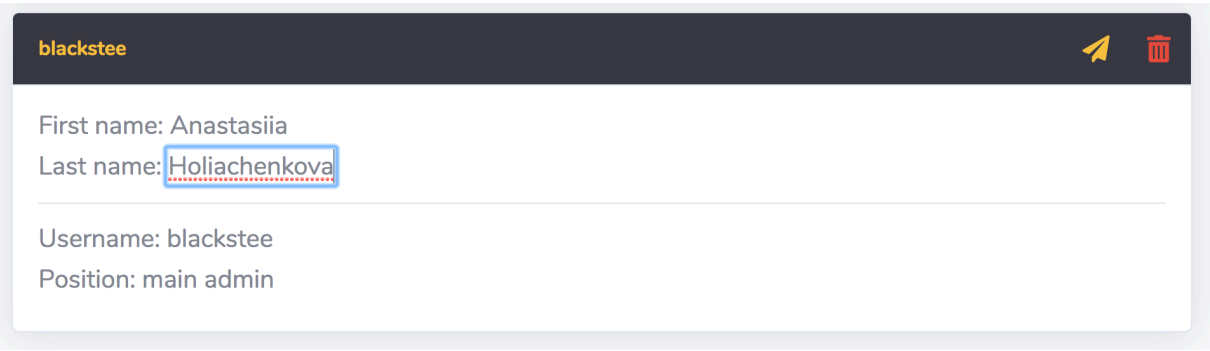


Рис. 23. Режим редагування користувача

4.4. Генерація звітів

Адміністратор може генерувати звіти із врахуванням деяких параметрів для журналу змін та користувачів. Для генерації звіту по користувачам він має натиснути кнопку “Report” на сторінці списку користувачів.

Аналогічно для генерації звіту по змінам у системі Адміністратор має натиснути кнопку “Report” на сторінці журналу змін.

Після генерації звіти автоматично завантажуються у форматі xls.

4.5. Генерація ієрархічної структури

Адміністратор може генерувати індивідуальну ієрархічну структуру відповідно до своєї компанії та посад у ній. Для цього він має натиснути кнопку “Dependencies tree” на сторінці списку користувачів. Після цього система відкриває сторінку генерації ієрархічної структури (див. рис. 24).

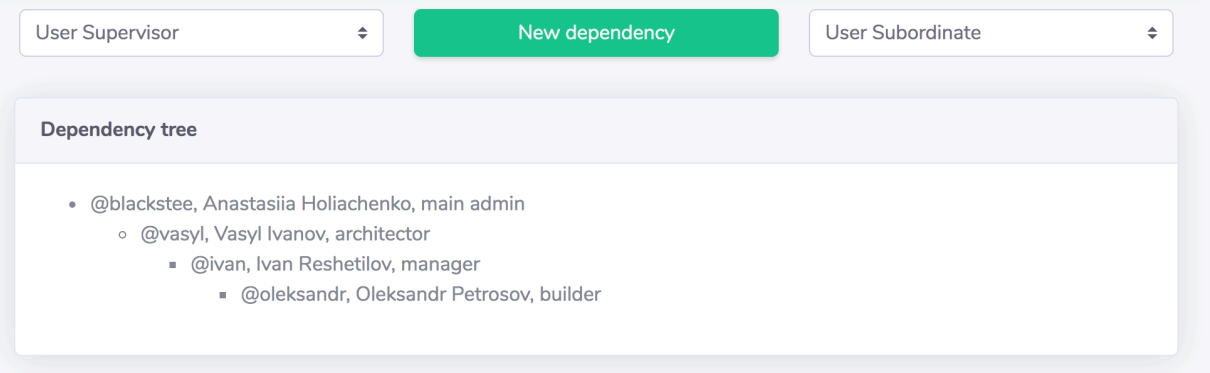


Рис. 24. Сторінка створення ієрархічної структури

5. РОБОТА ІЗ ДБН

5.1. Сторінка зі списком ДБН

Кожен користувач має доступ до сторінки зі списком ДБН (Державні Будівельні Норми). Для переходу на цю сторінку він має обрати “Rules DBN” на лівій панелі інструментів. На сторінці роботи із ДБН показаний список ДБН із офіційною назвою та номером (див. рис. 25).

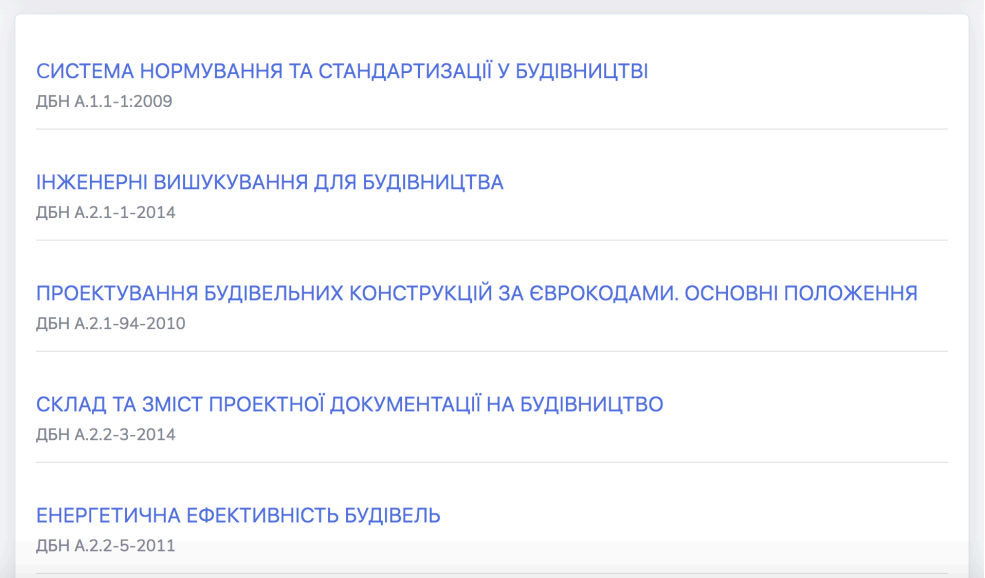


Рис. 25. Сторінка роботи зі списком ДБН

5.2. Сторінка із обраним ДБН

Для переходу до конкретного ДБН користувач має натиснути на нього, після чого у браузері завантажується файл у форматі pdf із обраним ДБН (див. рис. 26).

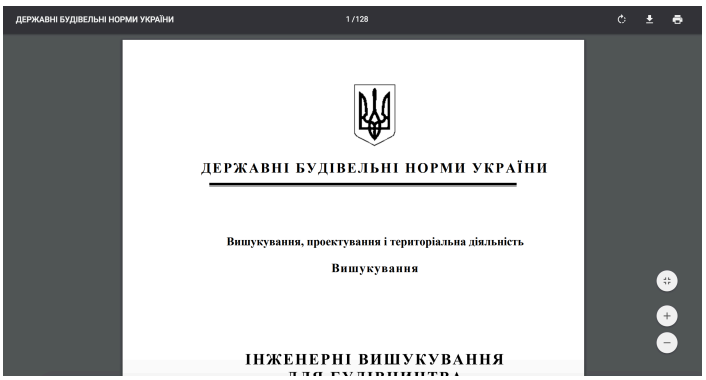


Рис. 26. Сторінка із обраним ДБН

6. РОБОТА ІЗ ЖУРНАЛОМ ПРОПОЗИЦІЙ

6.1. Внесення нової пропозиції

Кожен користувач може вносити пропозиції до журналу пропозицій. Для цього він має перейти на сторінку “Suggestions” та натиснути на кнопку “Add new suggestion” (див. рис. 27).

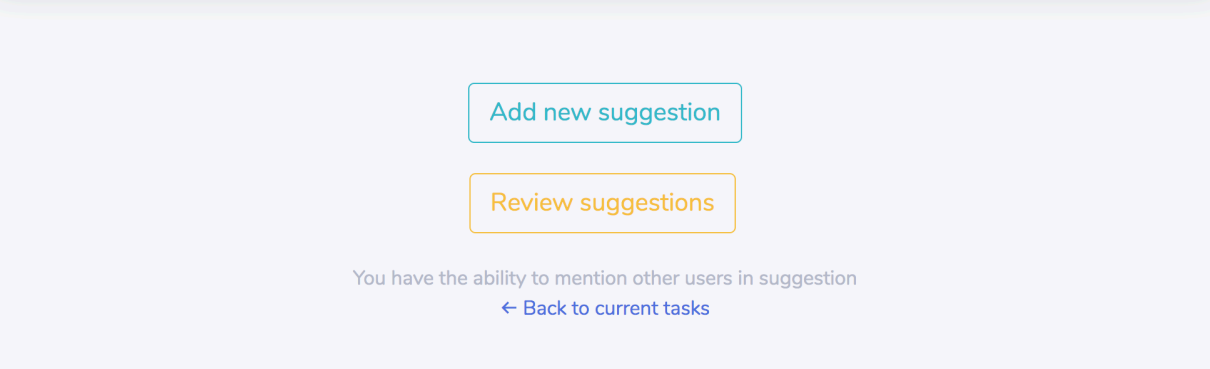


Рис. 27. Сторінка пропозицій

Після того як користувач заповнить поле пропозиції у відкритому модальному вікні створення пропозицій, (див. рис. 28) вона буде збережена.

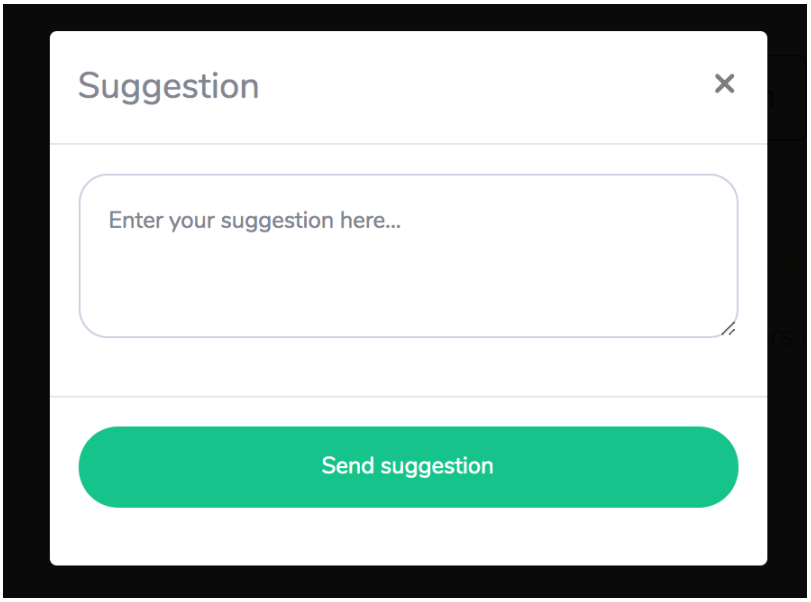


Рис. 28. Модальне вікно створення пропозиції

6.2. Сторінка із журналом пропозицій

Лише користувач типу “Менеджер” та “Топ-менеджер” має доступ до сторінки із журналом пропозицій. Для переходу на цю сторінку він має обрати “Review suggestions” на сторінці пропозицій (див. рис. 29).

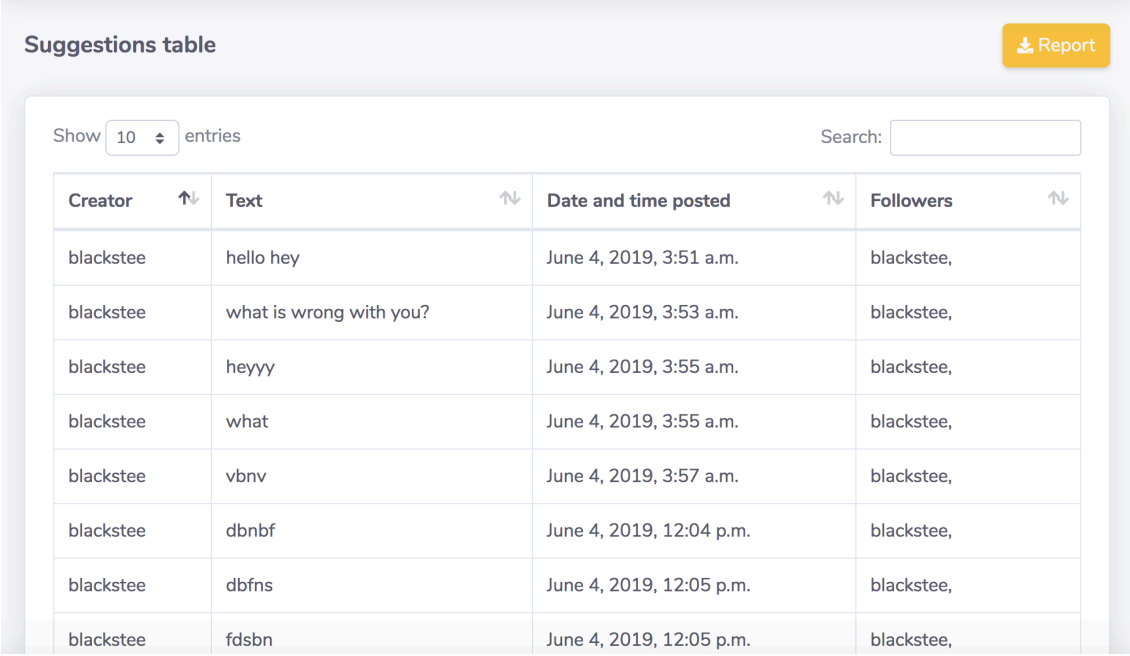


Рис. 29. Сторінка із журналом пропозицій

6.3. Генерація звіту по пропозиціям

Користувач типу “Менеджер” та “Топ-менеджер” може генерувати звіти із врахуванням параметрів для журналу пропозицій. Для генерації звіту він має натиснути кнопку “Report” на сторінці журналу пропозицій (див рис. 30). Після генерації звіт автоматично завантажується у форматі xls.

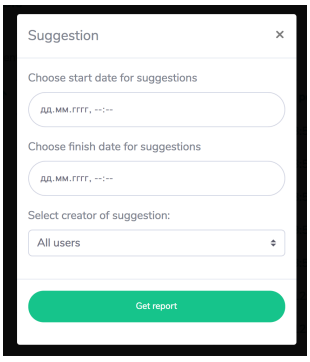


Рис. 30. Модальне вікно задання параметрів для звіту по пропозиціям